
PRIMOS® TCP/IP Guide

Third Edition

Bernard Gilman

*This guide documents the software operation
of the Prime Computer and its supporting
systems and utilities as implemented at
Master Disk Revision 22.0 (Rev. 22.0).*

Prime Computer, Inc., Prime Park, Natick, MA 01760

Copyright Information

The information in this document is subject to change without notice and should not be construed as a commitment by Prime Computer, Inc. Prime Computer, Inc. assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

Copyright © 1988 by Prime Computer, Inc., Prime Park, Natick, Massachusetts 01760

PRIME, PRIME, PRIMOS, and the PRIME logo are registered trademarks of Prime Computer, Inc. DISCOVER, EDMS, FM+, INFO/BASIC, INFORM, Prime INFORMATION, Prime INFORMATION CONNECTION, Prime INFORMATION EXL, MDL, MIDAS, MIDASPLUS, PRIME EXL, PRIME MEDUSA, PERFORM, PERFORMER, PRIME/SNA, PRIME TIMER, PRIMAN, PRIMELINK, PRIMENET, PRIMEWAY, PRIMEWORD, PRIMIX, PRISAM, PRODUCER, Prime INFORMATION/pc, PST 100, PT25, PT45, PT65, PT200, PT250, PW153, PW200, PW250, RINGNET, SIMPLE, 50 Series, 400, 750, 850, 2250, 2350, 2450, 2455, 2550, 2655, 2755, 4050, 4150, 6350, 6550, 9650, 9655, 9750, 9755, 9950, 9955, and 9955II are trademarks of Prime Computer, Inc.

Printing History

First Edition (DOC10155-1LA) July 1987 for Revision (IPR) 1 - 21.0

Second Edition (DOC10155-2LA) March 1988 for Revision (IPR) 1 - 21.0

Third Edition (DOC10155-3LA) October 1988 for Revision (IPR) 2 - 22.0

Credits

Editorial: Barbara Fowlkes

Design: Carol Smith

Project Support: Kim Haase, Jane Danielson

Graphics Support: Robert Stuart

Illustration: Michael Moyle

Document Preparation: Mary Mixon

Composition: Julie Cyphers, Sharon Temple

Production: Judy Gordon

How To Order Technical Documents

Follow the instructions below to obtain a catalog, a price list, and information on placing orders.

United States Only: Call Prime Telemarketing, toll free, at 800-343-2533, Monday through Friday, 8:30 a.m. to 5:00 p.m. (EST).

International: Contact your local Prime subsidiary or distributor.

Customer Support Center

Prime provides the following toll-free numbers for customers in the United States needing service:

1-800-322-2838 (Massachusetts)

1-800-541-8888 (Alaska and Hawaii)

1-800-343-2320 (within other states)

For other locations, contact your Prime representative.

Surveys and Correspondence

Please comment on this manual using the Reader Response Form provided in the back of this book. Address any additional comments on this or other Prime documents to:

Technical Publications Department
Prime Computer, Inc.
500 Old Connecticut Path
Framingham, MA 01701

About This Book	xi
------------------------	----

Part I: User

1 Introduction to PRIMOS TCP/IP	1-1
PRIMOS TCP/IP Architecture	1-1
LAN300	1-2
LAN Host Controller 300	1-4
PRIMOS TCP/IP User Facilities	1-4
Programmer Facilities	1-6
PRIMOS TCP/IP Administrative Facilities	1-6
Summary of Release 2.0 Enhancements	1-7
Restrictions and Limitations	1-8
2 User TELNET	2-1
Invoking TELNET on a 50 Series Host	2-1
TELNET Commands	2-4
3 Using PRIMOS FTP on a 50 Series System	3-1
Before You Use FTP	3-1
Access Rights	3-1
Submitting a Request	3-4
Format of FTP Commands	3-5
Invoking FTP	3-7
Selecting File Types	3-12
Sending Files	3-14
Retrieving Files	3-18
Executing Other Commands From Within FTP	3-22
FTP Help Facility	3-25
Quitting PRIMOS FTP	3-26
Summary of FTP Commands	3-26
Sample Session	3-29

4	A PRIMOS FTP Commands Reference	4-1
	Format of FTP Commands	4-1
	Local Commands	4-2
	PRIMOS FTP User Commands	4-6
5	Accessing PRIMOS FTP From a Remote System	5-1
	Before You Transfer a File	5-1
	Project IDs	5-3
	Transferring Files From Password-protected Directories	5-4
	FTP Commands Supported by the PRIMOS FTP Server	5-4
6	PRIMOS TCP/IP MAIL	6-1
	Before You Start	6-1
	Using PRIMOS TCP/IP MAIL	6-4
	PRIMOS TCP/IP MAIL Help Facility	6-11
	If Mail Cannot Be Delivered	6-11
7	A PRIMOS TCP/IP MAIL Commands Reference	7-1
	MAIL Commands at PRIMOS Command Level	7-1
	PRIMOS TCP/IP MAIL Interactive Commands	7-7

Part II: Programmer

8	Introduction to the PRIMOS Socket Library	8-1
	Prerequisites	8-1
	Socket Concepts	8-2
	Supported Socket Calls	8-5
	Usage	8-8
	Differences Between PRIMOS and UNIX	
	Socket Support	8-21
	Programming Considerations	8-27
	Restrictions and Limitations	8-30

9	Socket Subroutine Calls	9-1
	Summary of Routines	9-1
	Subroutine Descriptions	9-3
	Socket Environment Subroutine Calls	9-5
	TUP\$close_socket_env()	9-5
	TUP\$pass_socket()	9-7
	TUP\$receive_passed_socket()	9-9
	TUP\$reset_socket_env()	9-10
	TUP\$setup_socket_env()	9-13
	PRIMOS Socket Subroutine Calls	9-16
	TUP\$accept()	9-16
	TUP\$bind()	9-19
	TUP\$close_socket()	9-22
	TUP\$connect()	9-23
	TUP\$getpeername()	9-26
	TUP\$getsockname()	9-28
	TUP\$getsockopt()	9-30
	TUP\$ioctl()	9-33
	TUP\$listen()	9-35
	TUP\$perror()	9-37
	TUP\$recv(), TUP\$recvfrom()	9-38
	TUP\$select()	9-41
	TUP\$send(), TUP\$sendto()	9-43
	TUP\$setsockopt()	9-46
	TUP\$shutdown()	9-49
	TUP\$socket()	9-51
	Network Library Subroutine Calls	9-53
	TUP\$getalladdrs()	9-54
	TUP\$getbroadcastaddr()	9-55

TUP\$getdevaddrs()	9-57
TUP\$gethostbyaddr()	9-60
TUP\$gethostbyname() ,	9-63
TUP\$gethostname()	9-65
TUP\$inet_addr()	9-66
TUP\$inet_lnaof()	9-67
TUP\$inet_makeaddr()	9-68
TUP\$inet_netof()	9-69
TUP\$inet_network()	9-70
TUP\$inet_ntoa()	9-71
10 Sample Programs	10-1
Sample Program 1: Client Program	10-1
Sample Program 2: Server Program	10-4
Part III: Administrator	
11 Installing PRIMOS TCP/IP	11-1
Prerequisites	11-1
Installation Procedure	11-3
PRIMOS TCP/IP Directories and Files	11-7
Access Control List Settings	11-14
Maintaining System Search Rules	11-17
12 Configuring PRIMOS TCP/IP	12-1
Overview	12-1
Configuring PRIMOS TCP/IP	12-1
The HOSTS.TXT Configuration File	12-5
The THISHOST File	12-11
The HOSTNAME_CONFIG File	12-14
Host Table Utility	12-16
Modifying the System Configuration File	12-17

	Configuring Network Terminal Service for PRIMOS TCP/IP Users	12-19
	Configuring PRIMOS TCP/IP When NTS Is Installed	12-24
	Configuring DSM for PRIMOS TCP/IP Unsolicited Messages	12-24
13	Configuring PRIMOS TCP/IP MAIL	13-1
	MAIL Operations	13-1
	Required MAIL Configuration Parameters	13-4
	Optional Time and Transmission Parameters	13-7
	Optional Node and Routing Parameters	13-11
	Sample MAIL Configuration Files	13-13
	Modifying the MAIL Configuration File	13-18
	Creating Mailboxes	13-18
14	Activating PRIMOS TCP/IP	14-1
	PRIMOS TCP/IP Processes and Phantoms	14-1
	Starting PRIMOS TCP/IP	14-5
	Process and Phantom Names	14-12
	The Allocation of Sockets	14-13
	How to Update the HOSTS.TXT or HOSTNAME_CONFIG File	14-14
	Shutting Down PRIMOS TCP/IP	14-15
	Modifying the PRIMOS.COMI File	14-16
15	Sample PRIMOS TCP/IP Configurations	15-1
	Sample Network 1	15-1
	Sample Network 2	15-4
	Sample Network 3	15-7
	Sample Network 4	15-11
16	Monitoring PRIMOS TCP/IP	16-1
	LHC Polling and Recovery	16-1

Log Files	16-2
LIST_LHC_STATUS Command	16-4

Appendices

A	User TELNET Error Messages	A-1
B	PRIMOS FTP Program Messages	B-1
	User FTP Program Messages	B-1
	FTP Server Process Program Messages	B-9
C	MAIL Messages	C-1
D	PRIMOS TCP/IP Configuration and Installation Error Messages	D-1
	TCP/IPHTU Error Messages	D-1
	COMM_CONTROLLER Command Error Messages	D-2
E	Network Management and Monitoring Program Messages	E-1
F	NTS-related Configuration Directives	F-1
G	Socket Library Error Messages	G-1
H	PRIMOS TCP/IP Glossary	H-1
	Index	Index-1

About This Book

TCP/IP for PRIMOS® Systems (PRIMOS TCP/IP) enables a user on a 50 Series™ host to communicate with other systems running the Transmission Control Protocol and Internet Protocol (TCP/IP) over a LAN300, the Prime IEEE 802.3 compliant local area network.

This guide provides tutorial and reference information on the following topics:

- Establishing a login connection to a remote host with TELNET
- Transferring a file with the File Transfer Protocol (FTP)
- Sending messages with the Simple Mail Transfer Protocol (SMTP)
- Writing network applications with the PRIMOS socket subroutine library
- Installing, configuring, and starting PRIMOS TCP/IP
- Monitoring PRIMOS TCP/IP

This guide is intended for

- Users who want to log in to a remote system with TELNET, send files to a remote user with FTP, or send mail to a remote user with SMTP
- Programmers who want to write network applications
- Network Administrators and operators who have responsibility for maintaining PRIMOS TCP/IP

This guide contains 16 chapters and eight appendices. It is divided into three parts: User, Programmer, and Administrator. Terminology that appears in the glossary is printed in bold type the first time a term is introduced.

Part I: User

Chapter 1, Introduction to PRIMOS TCP/IP, is an overview of TCP/IP on a 50 Series system running PRIMOS.

Chapter 2, User TELNET, describes how to use TELNET to establish a remote login connection to a remote system.

Chapter 3, Using PRIMOS FTP on a 50 Series System, is a tutorial that describes how a user on a 50 Series system can send files to and retrieve files from a remote system.

Chapter 4, A PRIMOS FTP Commands Reference, provides a detailed description of all FTP commands for local 50 Series users.

Chapter 5, Accessing PRIMOS FTP From a Remote System, describes the FTP commands that PRIMOS FTP supports when a user on another vendor's system accesses a 50 Series system. Remote users should read this chapter to learn what FTP commands you can invoke on your local system to connect to a 50 Series host.

Chapter 6, PRIMOS TCP/IP MAIL, describes how to send mail messages to a remote system that supports the Simple Mail Transfer Protocol (SMTP).

Chapter 7, A PRIMOS TCP/IP MAIL Commands Reference, describes all the MAIL commands that TCP/IP MAIL supports.

Part II: Programmer

Chapter 8, Introduction to the PRIMOS Socket Library, describes the 4.3BSD socket concepts and describes the facilities that the PRIMOS socket library supports. The supported subroutine calls are described. Programming examples illustrate how to write a network application.

Chapter 9, Socket Subroutine Calls, is a reference chapter that describes all the subroutine calls that are supported.

Chapter 10, Sample Programs, includes two sample programs that illustrate how a network application (TCP) can call the socket library.

Part III: Administrator

Chapter 11, Installing PRIMOS TCP/IP, describes how to install PRIMOS TCP/IP on a 50 Series system. This chapter is intended for Network Administrators and operators.

Chapter 12, Configuring PRIMOS TCP/IP, describes how to create configuration files for PRIMOS TCP/IP. This chapter is intended for Network Administrators and operators.

Chapter 13, Configuring PRIMOS TCP/IP MAIL, describes to how to configure the PRIMOS TCP/IP MAIL product so that users can send and receive mail across a network. This chapter is intended for Network Administrators and operators.

Chapter 14, Activating PRIMOS TCP/IP, explains how to start PRIMOS TCP/IP software on a 50 Series host and LAN Host Controller 300 (LHC). This chapter is intended for Network Administrators and operators.

Chapter 15, Sample PRIMOS TCP/IP Configurations, describes how to configure four sample networks for TCP/IP. This chapter is intended for Network Administrators and operators.

Chapter 16, Monitoring PRIMOS TCP/IP, describes how to monitor TCP/IP. This chapter is intended for Network Administrators and operators.

Appendices

Appendix A, User TELNET Error Messages, describes the TELNET messages that can return to a local user.

Appendix B, PRIMOS FTP Program Messages, describes FTP messages that can return to local and remote users.

Appendix C, MAIL Messages, describes the error messages that TCP/IP MAIL can return to users on a 50 Series system.

Appendix D, PRIMOS TCP/IP Configuration and Installation Error Messages, describes the error messages that can appear when you install and configure TCP/IP on a 50 Series system.

Appendix E, Network Management and Monitoring Program Messages, describes messages that are displayed when a Network Administrator or operator monitors the operation of a LAN Host Controller 300.

Appendix F, NTS-related Configuration Directives, describes the system configuration directives that you must modify when you install NTS for PRIMOS TCP/IP.

Appendix G, Socket Library Error Messages, lists the socket library errors and error messages.

Appendix H, PRIMOS TCP/IP Glossary, contains definitions of terms that refer to PRIMOS TCP/IP, Prime networks, or data communications in general.

NTS Planning and Configuration Guide

Network Administrators and operators should also read the *NTS Planning and Configuration Guide* (DOC10159-11A). Network Terminal Service (NTS) is a prerequisite for running server TELNET. This guide describes the hardware requirements and the configuration file for the Network Terminal Service. The following chapters are particularly relevant to PRIMOS TCP/IP:

- Chapter 2, Installing NTS, describes how to install NTS software.
- Chapter 4, Configuring an NTS Network, describes the NTS configuration file and how to use the NTS configuration utility.

Related Documentation

Users may find the following Prime documentation helpful:

- *PRIMOS User's Guide* (DOC4130-5LA) is an introduction to the PRIMOS operating system.
- *PRIMOS Commands Reference Guide* (DOC3108-7LA) is a detailed reference of user commands.

Programmers may find the following Prime documentation helpful:

- *C User's Guide* (DOC7534-3LA)
- *Subroutines Reference V: Event Synchronization* (DOC10213-1LA)

Network Administrators and operators may find the following Prime documentation helpful:

- *DSM User's Guide* (DOC10061-1LA) describes how to configure and use Distributed Systems Management (DSM).
- *NTS User's Guide* (DOC10117-1LA) describes how to use the Network Terminal Service (NTS) to access 50 Series systems on a LAN300.
- *Operator's Guide to Prime Networks* (DOC10114-1LA) provides reference information on running network-related programs and monitoring network events.
- *Operator's Guide to System Commands* (DOC9304-3LA) describes the PRIMOS commands that an operator can invoke at the supervisor terminal.
- *Operator's Guide to File System Maintenance* (DOC9300-3LA) describes the PRIMOS File System.
- *PRIMENET Planning and Configuration Guide* (DOC7532-1LA) describes how to install PRIMENET™ and the File Transfer Service (FTS).
- *System Administrator's Guide, Volume I: System Configuration* (DOC10131-1LA) describes how a System Administrator can plan and configure a Prime computer system and users' environments.
- *System Administrator's Guide, Volume II: Communication Lines and Controllers* (DOC10132-1LA) describes how a System Administrator can configure local and network configuration lines, and provides an overview of AMLC, ICS, and LHC300 controllers.
- *System Administrator's Guide, Volume III: System Access and Security* (DOC10133-1LA) and its update package (UPD10133-11A) describe how a System Administrator can establish and maintain system access and security.
- *Translator Family Software Release Document, Release T1.0-21.0* (DOC10217-1PA) describes the installation procedure for Prime Translator products, libraries, and utilities.

Network programmers should consult the following documents, which describe the 4.3BSD version of sockets:

- *Berkeley Software Architecture Manual, 4.3BSD Edition*
- *Programmer's Reference Manual*
- *UNIX Programmer's Guide*

The Berkeley Software Architecture Manual is published by the Computer Systems Research Group, University of California, Berkeley, California 94720.

Network Administrators may find useful the following publication, which describes the standard Department of Defense military protocols: *DDN Protocol Handbook*, Volumes One, Two, and Three, published by the DDN Network Information Center SRI International, Menlo Park, CA.

The following chart shows where to find the information that you need in the Prime networks document set. There is a column for each network product and a row for each functional activity. For example, to find information on monitoring and controlling NTS, refer to the *Operator's Guide to Prime Networks*.

	PRIMENET	NTS	TCP/IP
Configuration	PRIMENET Planning and Configuration Guide	NTS Planning and Configuration Guide	
Installation		LTS300 Installation Guide	PRIMOS TCP/IP Guide
Monitoring and Control	Operator's Guide to Prime Networks		
Using the Software	User's Guide to Prime Network Services	NTS User's Guide	
Programming	Programmer's Guide to Prime Networks		

Prime Documentation Conventions

The following conventions are used in command formats, statement formats, and in examples throughout this document. Examples illustrate the uses of these commands and statements in typical applications.

<i>Convention</i>	<i>Explanation</i>	<i>Example</i>
UPPERCASE	In command formats, words in uppercase bold indicate the names of commands, options, statements, and keywords. Enter them in either uppercase or lowercase.	SLIST
<i>italic</i>	In command formats, words in lowercase bold italic indicate variables for which you must substitute a suitable value. In text and in messages, variables are in non-bold lowercase italic.	LOGIN <i>user-id</i> Supply a value for <i>x</i> between 1 and 10.

<i>Convention</i>	<i>Explanation</i>	<i>Example</i>
Abbreviations in format statements	If a command or option has an abbreviation, the abbreviation is placed immediately below the full form.	SET_QUOTA SQ
Brackets []	Brackets enclose a list of one or more optional items. Choose none, one, or several of these items.	LD [-BRIEF -SIZE]
Braces { }	Braces enclose a list of items. Choose one and only one of these items.	CLOSE {filename ALL}
Braces within brackets [{}]	Braces within brackets enclose a list of items. Choose either none or only one of these items; do not choose more than one.	BIND [{pathname options}]
Vertical bars 	Vertical bars enclose a list of items. Choose one or more of these items.	OUTPUT filename TTY
Parentheses ()	In command or statement formats, you must enter parentheses exactly as shown.	DIM array (row, col)
User input in examples	In examples, user input is in bold italic but system prompts and output are not.	OK, <i>RESUME MY_PROG</i> This is the output of MY_PROG.CPL OK,
Ellipsis ...	An ellipsis indicates that you have the option of entering several items of the same kind on the command line.	SHUTDN <i>pdev-1</i> [...pdev-n]
Hyphen -	Wherever a hyphen appears as the first character of an option, it is a required part of that option.	SPOOL -LIST
Default indicator •	In a list of options, a bullet indicates the default choice, if one exists. If you do not select an option, the system chooses the default option.	{ A • O D }
Subscript	A subscript after a number indicates that the number is not in base 10. For example, the subscript 8 is used for octal numbers.	200 ₈
Key symbol	In examples and text, the name of a key enclosed by a rectangle indicates that you press that key.	Press Return

Introduction to PRIMOS TCP/IP

TCP/IP for PRIMOS Systems (PRIMOS TCP/IP) is a Prime communications product that enables a 50 Series host to communicate with either a Prime or another vendor's workstation or system running the Transmission Control Protocol (TCP) and the Internet Protocol (IP) over a LAN300, the Prime IEEE 802.3 compliant Local Area Network. At Release 2.0, PRIMOS TCP/IP implements TCP, IP, UDP, the File Transfer Protocol (FTP), user and server TELNET, the Simple Mail Transfer Protocol and a subset of Berkeley UNIX 4.3BSD socket library subroutine calls. The TELNET protocol enables a user on a 50 Series host to log in to a remote host and enables a user on another system connected to the LAN300 to log in to a 50 Series host. SMTP defines the format of mail messages and is the protocol by which they are exchanged.

Note

Release 1.0 of PRIMOS TCP/IP was called Workstation/System Interconnect 300 (WSI300). PRIMOS TCP/IP is an enhancement of WSI300.

PRIMOS TCP/IP runs on a 50 Series host and the Prime LAN Host Controller 300 (LHC) over an IEEE 802.3 coaxial cable. The LHC is an intelligent communications controller that enables the 50 Series host to offload protocols.

This chapter describes PRIMOS TCP/IP

- Architecture
- User facilities
- Operator facilities
- Restrictions and limitations

PRIMOS TCP/IP Architecture

TCP/IP belongs to a standard set of protocols that were originally developed by the Department of Defense (DoD) for use in military wide area networks and then adapted for use in local area networks.

The Internet Protocol (IP) is a datagram protocol that puts data into packets, routes the packets to their Internet addresses, and reassembles the packets into data at their destination. IP provides

communications support for upper layer protocols such as the Transmission Control Protocol and the User Datagram Protocol. IP provides a service that corresponds to layer 3 (Network) of the International Organization for Standardization (ISO) Reference Model for Open Systems Interconnection (OSI Reference Model). See Table 1-1.

Table 1-1
PRIMOS TCP/IP and the ISO/OSI Reference Model

<i>OSI Reference Model</i>	<i>PRIMOS TCP/IP</i>
7 Application	FTP/TELNET/SMTP
6 Presentation	FTP/TELNET/SMTP
5 Session	FTP/TELNET/SMTP
4 Transport	TCP/UDP
3 Network	IP
2 Data Link	Ethernet™
1 Physical	IEEE 802.3 CSMA/CD

The Transmission Control Protocol (TCP) is the higher level transmission protocol that corresponds to layer 4 (Transport) of the ISO/OSI Model. TCP is a reliable connection-oriented transport protocol. Packet sequencing, checksums, and retransmissions are used to maintain reliability.

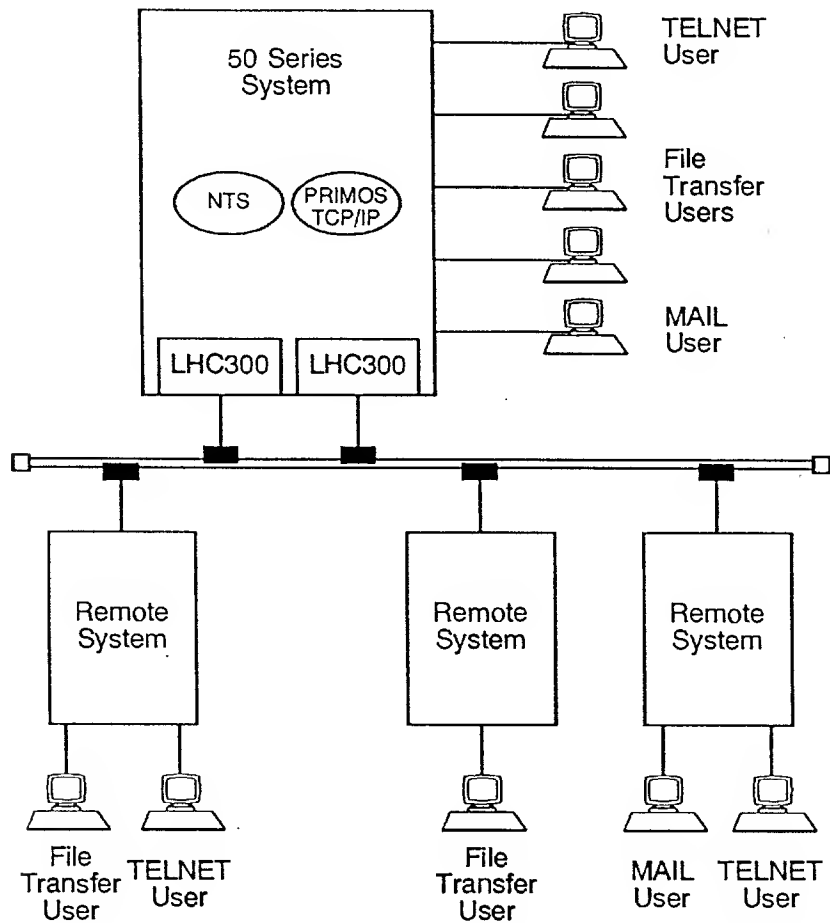
The User Datagram Protocol (UDP) is a transaction-oriented protocol that supports the transmission of unsequenced segments of data to an Internet destination system. UDP best serves upper-layer protocols (such as the Simple Mail Transfer Protocol) for which performance is paramount. UDP neither prevents duplication or guarantees end-to-end delivery.

The File Transfer Protocol (FTP), the Simple Mail Transfer Protocol (SMTP), and TELNET, the remote terminal access protocol, are applications that run on top of TCP or UDP (layers 5, 6, and 7 of the ISO/OSI Reference Model). FTP transfers files between computer systems. TELNET is the terminal communications facility that connects a local system to another system on the LAN300 by emulating a standardized terminal to which both systems can connect. SMTP is the communications protocol for transferring messages. The TELNET program and the MAIL program provide the user interface to the TELNET protocol and SMTP, respectively.

PRIMOS TCP/IP conforms to MIL STD 1778 for TCP, MIL STD 1777 for IP, MIL STD 1780 for FTP, MIL STD 1781 for SMTP, and MIL STD 1782 for TELNET.

LAN300

The LAN300, the Prime Local Area Network (LAN) in a bus topology, provides the method of transmitting data between systems. It corresponds to layers 1 and 2 (Physical and Data Link) of the ISO/OSI Reference Model. Refer to Figure 1-1.



Q10155-3LA-2.2

Figure 1-1
PRIMOS TCP/IP LAN300 Configuration

Layer 1, the physical interface, conforms to the IEEE 802.3 standard and employs the **Carrier Sense Multiple Access With Collision Detection (CSMA/CD)** access method. The CSMA/CD protocols enable nodes to share the communications channel. CSMA/CD protocols service each message on the LAN and provide terminals the opportunity to communicate as needed. Layer 2 (Data Link) is Ethernet.

A typical standard LAN300 is composed of bus segments, each of which can be as long as 500 meters. Devices are connected to the LAN at a station. A station is a point on the LAN at which a controller, transceiver, and tap are attached to the bus segment. A station supports a Prime or another vendor's host, and a cluster of terminals or serial printers, or both. A bus segment can have a maximum of 100 stations. A fully configured LAN can have a maximum of 1024 stations.

Chapter 6, PRIMOS TCP/IP MAIL, describes how to use MAIL. Chapter 7, A PRIMOS TCP/IP MAIL Commands Reference, describes all MAIL commands. Chapter 13, Configuring PRIMOS TCP/IP MAIL, explains how a Network Administrator configures MAIL so that users can send and receive messages across a network.

Programmer Facilities

PRIMOS TCP/IP supports a subset of the Berkeley UNIX 4.3BSD (Berkeley Software Distribution) socket subroutine calls. A network application that calls the socket library can access the TCP or UDP protocols running on a LHC300. This enables an application running on a 50 Series host to communicate with an application running on a remote host. A user program that calls the socket library can use the PRIMOS InterServer Communications (ISC) facility to communicate with another application running on the same system or running over PRIMENET. New PRIMOS socket subroutine calls are provided to establish the socket environment on a 50 Series host with LHC300 controllers.

Chapter 8, Introduction to the PRIMOS Socket Library, and Chapter 9, Socket Subroutine Calls, describe the Primc implementation of sockets.

PRIMOS TCP/IP Administrative Facilities

A Network Administrator or operator can install and configure PRIMOS TCP/IP. After installing PRIMOS TCP/IP, a Network Administrator or operator creates three ASCII text files with a text editor (such as EMACS or ED) to configure the network.

The first file (named HOSTS.TXT) specifies the names and addresses of local and remote hosts and networks. The second configuration file (named THISHOST) identifies the local node. The third configuration file (named HOSTNAME_CONFIG) is required by the Internet Hostname Service protocol. (The Hostname Service protocol enables a 50 Series system to use a TCP connection to retrieve a remote host name or Internet address over the network.)

After these files are created, the Network Administrator or operator runs a utility program to prepare the information in these files for PRIMOS TCP/IP.

The Network Administrator must also add the LHC directive to the system configuration file and configure the Network Terminal Service (NTS) for PRIMOS TCP/IP. How to use the CONFIG_NTS utility is described in the *NTS Planning and Configuration Guide*.

A Network Administrator or operator can perform the following tasks with PRIMOS TCP/IP:

- Downline load the LAN300 controller board with appropriate PRIMOS TCP/IP files
- Configure, start, and stop the network without rebooting the host system
- Update the HOSTS.TXT and HOSTNAME_CONFIG configuration files without stopping and restarting TCP/IP
- Add and remove nodes from the network without disrupting users on the network

Network Management

PRIMOS TCP/IP shares network management facilities with the Network Terminal Service (NTS) and PRIMENET. PRIMOS TCP/IP offers the following network management features:

- Automatic detection and recovery if the LHC fails
- Network status information to Network Administrators, operators, and users
- Isolation of network faults, as provided through diagnostics in the LHC host controller board

Chapter 16 describes how to monitor PRIMOS TCP/IP.

Summary of Release 2.0 Enhancements

Release 2.0 of PRIMOS TCP/IP supports the new features described in the next sections.

New TELNET Support

A local user can now log in to a remote system with User TELNET.

Chapter 2, User TELNET, describes the features of TELNET.

New User FTP Support

User FTP is implemented with PRIMOS socket library routines. The WSIFTP_USER0 and WS1_USER_PHANTOM nn processes and associated startup commands have been eliminated.

When you open a connection to a remote host, you can enter either an Internet address or a host name. After you open a connection to a remote host, FTP now displays your local user ID as a default remote login.

FTP also supports EDIT_CMD_LINE (ECL), the Prime command line editor. ECL allows you to edit, save, and redisplay FTP commands and arguments that you enter.

You can now execute PRIMOS commands from within FTP.

FTP also supports the MGET and MPUT commands. MGET enables you to retrieve multiple files. MPUT enables you to send multiple files. MGET and MPUT support wildcard expansions.

Chapter 3, Using PRIMOS FTP on a 50 Series System, describes these new FTP features.

SMTP (MAIL)

With MAIL, you can send messages to other users on other hosts that support SMTP.

Chapter 6, PRIMOS TCP/IP MAIL, describes how to use MAIL.

Sockets

Programmers can write programs that call PRIMOS socket library routines to access UDP and TCP protocols.

Chapter 8, Introduction to the PRIMOS Socket Library, describes PRIMOS socket support.

New Administrator Features

PRIMOS TCP/IP supports the client portion of the Hostname Service protocol. A 50 Series system can retrieve a host name or an Internet address from the host table of a system that supports the server portion of the Hostname Service protocol.

A System Administrator can now update the HOSTS.TXT file without stopping and starting TCP/IP.

Restrictions and Limitations

At Release 2.0 of PRIMOS TCP/IP, the following restrictions and limitations apply:

- PRIMOS TCP/IP requires PRIMOS Rev. 22.0 or later.
- Workstation/System Interconnect 300 (WSI300) runs only on PRIMOS Rev. 21.0.1 or later revisions of Rev. 21.
- PRIMOS TCP/IP provides FTP, and TELNET support only over a LAN300. PRIMOS TCP/IP does not provide support over X.25, 1822 links (ARPANET link protocol), or over any other media. MAIL provides service over all PRIMENET media including PRIMENET ring and public data networks.
- A 50 Series host running PRIMOS TCP/IP cannot serve as an IP gateway node between two LAN300s. That is, a Prime host cannot accept and retransmit data packets from a system on one LAN300 that are addressed to a system on another LAN300. The Prime host, however, can communicate with a system that can be reached only by a gateway.
- When two LHCs running PRIMOS TCP/IP on the same host are connected to the same LAN300; automatic load balancing is not provided.
- Files can be transferred only between a local system and another system on the LAN. One of these systems must be a local 50 Series host. The other system can be either a

Prime or another vendor's system. For example, a user on local system A cannot transfer files between system B and system C.

- PRIMOS TCP/IP can support a maximum of 32 FTP, user TELNET, or socket users simultaneously for each LHC.
- A process that calls the PRIMOS socket library can use a maximum of 32 sockets simultaneously for each LHC.
- PRIMOS FTP transfers a file as a DAM file.

User TELNET

User TELNET enables a user on a 50 Series system to establish a remote login connection to either a remote Prime system or another vendor's system.

Note

A user cannot open more than one TELNET connection simultaneously.

The user TELNET facility is implemented entirely on the local 50 Series system. The remote host to which you connect must support a server TELNET facility that operates over an IEEE 802.3 LAN.

See Appendix A for a description of TELNET error messages.

Invoking TELNET on a 50 Series Host

The TELNET command invokes the user TELNET facility from PRIMOS command level. The TELNET command has the following format:

```
TELNET [ host [port] ]  
        -HELP
```

You can enter the TELNET command in uppercase, lowercase, or a combination of both. The *host* argument can be either a host name or an Internet Address. The *port* argument is valid only in conjunction with the *host* argument and identifies an endpoint within a remote system to which you want to connect. If you omit the port number, TELNET selects the default port number.

If you issue the command without any arguments, you are placed in **Command mode**. Command mode displays the TELNET prompt as shown in the following example:

```
OK, TELNET
```

```
[TELNET Rev. 22.0 Copyright (c) 1988, Prime Computer Inc.]  
telnet>
```

In Command mode, the TELNET facility accepts and executes the TELNET commands in the following section.

When you execute the TELNET command with either the *host* or the *host* and *port* argument, TELNET opens a connection to the remote system. You are placed in **Input mode**. In Input mode, any data you enter is sent to the destination system.

In the following example, you want to connect to a system named SYS1:

OK, **TELNET SYS1**

[TELNET Rev. 22.0 Copyright (c) 1988, Prime Computer Inc.]

Entering INPUT mode. Your escape character is: ^]

Please wait...

Connecting to SYS1(188.31.2.1)

LOGIN JACK

In the following example, you want to connect to a remote host that has an Internet address of 193.44.5.2:

OK, **TELNET 193.44.5.2**

[TELNET Rev. 22.0 Copyright (c) 1988, Prime Computer Inc.]

Entering INPUT mode. Your escape character is: ^]

Please wait...

Connecting to 193.44.5.2

LOGIN JACK

If TELNET can associate the Internet address that you enter with a host name (listed in the HOSTS.TXT configuration file), TELNET displays both the host name and its Internet address in the Connecting to message. If TELNET cannot map the Internet address to a host name, TELNET displays only the Internet address, as in the above example.

When TELNET informs you that you are in Input mode, you can enter any commands required to log in to the remote system (for example, LOGIN JACK) .

If TCP/IP Has Not Been Started: If PRIMOS TCP/IP has not been started, you receive the following message:

The TCP/IP product has not been started.
It must be started by your system administrator
before the USER Telnet facility can be invoked.

Exiting TELNET

You automatically exit the TELNET program when

- You enter the TELNET command with a host name or Internet address, establish a connection, and then you log out of the remote host
- You enter the TELNET command with a host name or Internet address and the remote system terminates your login

In the following example, TELNET cannot establish a TCP/IP connection to the remote host or the TCP/IP connection has been broken:

OK, *TELNET SYS1*

[TELNET Rev. 22.0 Copyright (c) 1988, Prime Computer Inc.]

Entering INPUT mode. Your escape character is: ^]

Please wait...

Connecting to SYS1(188.31.2.1)

Remote not responding, still trying...
Remote not responding, still trying...
Remote not responding, still trying...
Remote not responding, still trying...
Remote not responding, still trying...
Remote not responding, still trying...

When TELNET continues to redisplay the message Remote not responding, still trying..., TELNET is attempting to establish a connection. You can wait, enter the escape character (described below), or enter **Ctrl** **P**. If you enter the escape character, TELNET attempts to return you to Command mode. If you enter **Ctrl** **P**, TELNET gives you the option of exiting TELNET to PRIMOS command level, terminating the current TELNET session, or continuing the attempt to establish the connection.

Appendix A describes the error messages that TELNET can display.

Escape Character Sequence

You cannot execute a TELNET command while in Input mode; you must switch to Command mode. To switch into Command mode, you must enter an **escape character**. An escape character is a control character combined with another character.

When you first enter TELNET, the enabled default escape character is the control key and the right bracket (**Ctrl** **]**). To specify this default escape character, hold down **Ctrl** and press **]**. You can change the default escape character with the ESCAPE command, described in the next section. Entering an escape character while you are in Command mode has no effect. After TELNET executes your command, you are switched back into Input mode. Commands that require options prompt you for an argument if you do not enter one. (Exceptions to this are entering the CLOSE and QUIT commands, described in the next section.)

Help Option

When you invoke the TELNET command with the **-HELP** option, the system displays syntax information. After the HELP text appears, you exit TELNET. In the following example, a user enters the TELNET command with the **-HELP** option:

```
OK, TELNET -HELP
```

```
TELNET [host] [port]
  host: Host name or address to connect to.
  port: Port to connect to.
```

```
OK,
```

TELNET Commands

You can enter the TELNET commands described in this section only when you are in Command mode. You can abbreviate commands. Most commands can be abbreviated as a single character. You can enter commands in uppercase, lowercase, or a combination of both.

Note

When you are in Command mode, the PRIMOS command line editor, **EDIT_CMD_LINE (ECL)**, is enabled. ECL allows you to edit, save, and redisplay any TELNET commands that you enter. If you have customized your environment with ECL, that environment is preserved. If you have disabled ECL with the **ECL -OFF** command, TELNET enables the ECL default characteristics. For more information on ECL, see the *PRIMOS Commands Reference Guide*.

After you connect to a remote system, you are in Input mode. While you are in Input mode, anything that you enter at your local terminal is transmitted as data to the remote host. To switch to Command mode, enter an escape character.

The following list summarizes TELNET commands:

ESCAPE E	Changes the TELNET default character
OPEN [host [port]] O	Opens a connection to a remote host
CLOSE C	Closes a connection
QUIT Q	Closes a connection and exits TELNET
DEBUG D	Toggles viewing of raw data through TELNET
HELP [command] H	Displays help information on a specific command
STATUS S	Shows the current status of the current TELNET connection
OPTIONS OP	Toggles viewing of TELNET options

ESCAPE Command

The ESCAPE command changes the TELNET default escape character. The default escape character is . An escape character lets you switch from Input to Command mode. An escape character has no effect when you are in Command mode.

The ESCAPE command has the following format:

ESCAPE

Enter the ESCAPE command without any arguments. TELNET prompts you to enter a new escape character if you want to change the default setting. The STATUS command described below displays your current escape character.

Valid escape characters are

a ... z

A ... Z

^

\

_

In the following example, a user enters the **ESCAPE** command without an argument and is prompted to enter a new escape character.

```
telnet> ESCAPE
```

```
      Please enter new escape character  
```

```
The new escape character is '^['.
```

```
Returning to INPUT mode.
```

OPEN Command

The **OPEN** command opens a connection to a remote system that you name. The **OPEN** command has the following format:

OPEN [*host* [*port*]]

host is the name or Internet address of the remote host to which you want to establish a connection. *port* is the port number of a process on the remote system. If you omit *host*, TELNET prompts you to enter one. If you do not specify a port number, TELNET uses a default number (23). If you specify a port number, you must specify a host name.

Once a connection is opened, you are in Input mode. If you specified a system name or Internet address on the command and you log out of the remote system, you are returned to Command mode. You can open a new connection.

CLOSE Command

The **CLOSE** command closes a connection. After you execute this command, you are in Command mode. You can open a new connection.

The following example shows the screen output of the **CLOSE** command:

```
telnet> CLOSE
```

```
telnet>
```

QUIT Command

The QUIT command closes a connection and exits TELNET.

The following example shows the screen output when you enter the QUIT command:

```
telnet> QUIT
```

```
Bye....
```

```
OK,
```

OPTIONS Command

The OPTIONS command toggles viewing TELNET protocol option negotiation. When the command is switched on, TELNET displays option negotiation. Options that the local node sends are displayed as **SENT**; options received from the remote node are displayed as **RCVD**.

The STATUS command tells whether option processing is switched on or off.

DEBUG

The DEBUG command toggles viewing of raw data through TELNET. When DEBUG is toggled on, TELNET displays all received data in hexadecimal and ASCII format.

The STATUS commands tells you whether the DEBUG command is switched on or off.

HELP Command

The HELP command provides help information on a specific TELNET command. The HELP command has the following format:

HELP [*command*]

command is the name of the command about which you want information.

Issuing the HELP command with no argument displays a list of supported TELNET commands. After help options are displayed, you are in Command mode so that you can request help on a specific command. If you are connected to a remote system and want to reenter Input mode without entering another HELP command, enter Return.

STATUS

The STATUS command shows the current status of the TELNET connection. STATUS returns the name and port of the peer system to which your local host is connected. The TELNET escape character is also displayed.

The following example shows the screen output of the STATUS command:

```
telnet> STATUS
```

```
Escape character is '^]'.
```

```
Connected to:
```

```
REMOTE_SYS.
```

```
Internet address is 129.122.3.50
```

```
Port number is 23
```

```
Echoing is being performed remotely
```

```
Debug is toggled off.
```

```
Option negotiation display is toggled off.
```

```
Returning to INPUT mode.
```

Using PRIMOS FTP on a 50 Series System

This chapter describes how to use the PRIMOS File Transfer Protocol (FTP) on a 50 Series system. FTP enables you to transfer files between a 50 Series system and a remote Prime or another vendor's system.

Chapter 4, A PRIMOS FTP Commands Reference, is a full reference of FTP commands for 50 Series users. Chapter 5, Accessing PRIMOS FTP From a Remote System, describes for remote users the FTP commands that the PRIMOS TCP/IP server process supports. Appendix B, FTP Program Messages, lists FTP error or program messages that may be displayed when you attempt to send a file to, or retrieve a file from, a remote system.

Before You Use FTP

Before you can use FTP, the following requirements must be met:

- PRIMOS TCP/IP must be installed on your system.
- An FTP product must be installed on the remote system.
- The remote system must be running.
- The LAN300 communication link between systems must be active.

Note

Because FTP does not use a queuing mechanism, you cannot submit a file transfer request (either interactively or by using a CPL file) when the communications link between the 50 Series system and the remote system is not operational or when the remote system is down.

Access Rights

Users of FTP are subject to the same ACL security mechanisms as other PRIMOS users and phantoms. Therefore, for FTP to work correctly, you need certain file access rights.

Note

Release 1.0 of WSI300 required that both local users and the ACL group .WSI_FTP\$ have the correct access rights to files and directories. Release 2.0 of PRIMOS TCP/IP (user FTP) requires only that local users have these rights.

Access rights are any combination of the following rights:

<i>Access Right</i>	<i>Description</i>
Owner (O)	An owner can set all rights (except P and ALL) on a file or a directory.
Protect (P)	Access rights on directories can be changed.
Delete (D)	Directory entries can be deleted.
Add (A)	Directory entries can be added.
List (L)	Directory contents can be listed.
Use (U)	A user can attach to a directory and use the directory name in a pathname.
Read (R)	A file can be read.
Write (W)	A file can be modified (overwritten).
Execute (X)	A local Executable Program Format (EPF) file can be executed but cannot be copied with the standard file system utilities.

Before you run FTP, check that you have the appropriate access rights, as explained in the next sections. See the *PRIMOS User's Guide* for information on setting ACLs in general.

Access Rights for the User on the 50 Series System

Before you can send or receive a file, FTP checks to see that you have access on the local host to the file that you want to transfer.

To send one or more files, you need the following rights on the 50 Series system:

- Read (R) access to the file or files
- Add (A), Delete (D), List (L), and Use (U) access to the user's current local directory
- Use (U) access to any parent directories

To retrieve one or more files from a remote host and store them on a 50 Series system, you need the following rights:

- Read (R) access to a temporary file that FTP creates when you retrieve multiple files
- Add (A), Delete (D), List (L), and Use (U) access to the directory on the 50 Series system where the file or files will be stored
- Use (U) access to any parent directories

Table 3-1 summarizes the access rights that a user on a 50 Series system needs to send files to and receive files from a remote system.

Table 3-1
Access Rights for 50 Series Users

<i>Access Right</i>	<i>Operation</i>
ADLUR	Send to remote system
ADLUR	Retrieve from remote system

Note

ADLUR access rights allow you to execute all PRIMOS FTP commands. Table 3-3, at the end of this chapter, lists the ACL rights that you need to execute specific FTP commands.

Transferring Files to a Remote 50 Series System: If the remote system is a 50 Series system, both the .TCP_FTP\$ ACL group and you (as a remote user) need access rights on that system to store a file or retrieve a file from one of its directories.

- You and the .TCP_FTP\$ ACL group need Add (A) and Use (U) access on the remote 50 Series system to store a file on it. You and the .TCP_FTP\$ ACL group need Read (R) and Write (W) access on the remote 50 Series system if, as part of storing a file, FTP needs to overwrite an existing file.
- You and the .TCP_FTP\$ ACL group need Read (R) access for the file and Use (U) access on a directory to retrieve a file from one of the remote 50 Series system's directories.
- To list the contents of a parent directory on a remote 50 Series system, both you and the .TCP_FTP\$ ACL group need List (L) and Use (U) access to that directory.
- If you want to delete files on the remote 50 Series system, both you and the .TCP_FTP\$ ACL group need Delete (D) and Use (U) access.

Check with the System Administrator of the remote 50 Series system to ensure that you have the access rights that you need. For more information on the .TCP_FTP\$ ACL group, see Chapter 5, Accessing PRIMOS FTP From a Remote System.

Checking Access Rights

Before you set or change access rights on the source directory on the 50 Series system, use the `LIST_ACCESS` command to list the access rights on that directory. In the following example, a 50 Series system user with the user ID JILL wants to send a file from a directory of a user with the user ID FRED. FRED is a user on the same 50 Series system. FRED checks JILL's access to FRED's directory:

```
OK, LIST_ACCESS
```

```
ACL protecting "<Current directory>":  
    FRED:      ALL  
    $REST:     NONE
```

In the example, user JILL has NONE access rights to the source directory on the 50 Series system. Therefore, user JILL cannot send files from FRED's directory. (The minimum access rights that JILL needs are R access to the file in FRED's directory and U access to FRED's directory.)

FRED can use the `EDIT_ACCESS` command to give JILL access to his directory.

```
OK, EDIT_ACCESS <SYS1>FRED JILL:RU  
OK, LIST_ACCESS
```

```
ACL protecting "<Current directory>":  
    FRED:      ALL  
    JILL:      RU  
    $REST:     NONE
```

JILL can now send a file from FRED's directory with the `SEND` command, which is described in a later section. RU access is sufficient to execute the `SEND` command but RU access is restrictive. JILL can send a file but she cannot list the contents of JACK's directory or check her access rights to that directory. JILL needs L access to perform these operations.

See Table 3-3, at the end of this chapter, for the access rights that each command requires.

Submitting a Request

To use FTP, you need to know the following information:

- Your PRIMOS user ID and password on the local 50 Series host
- A user ID and password on the remote system
- The name of the remote system
- The pathname of any files that you want to send or retrieve
- The file type of the files that you want to transfer

To send or retrieve a file, you must perform the following steps:

1. Log in to PRIMOS.
2. Invoke FTP.
3. Connect to the remote system.
4. Log in to the remote FTP server.
5. Enter FTP commands.
6. Exit FTP.

Figure 3-1 shows the sequence of steps in an FTP session. The sections that follow explain these steps in detail.

Format of FTP Commands

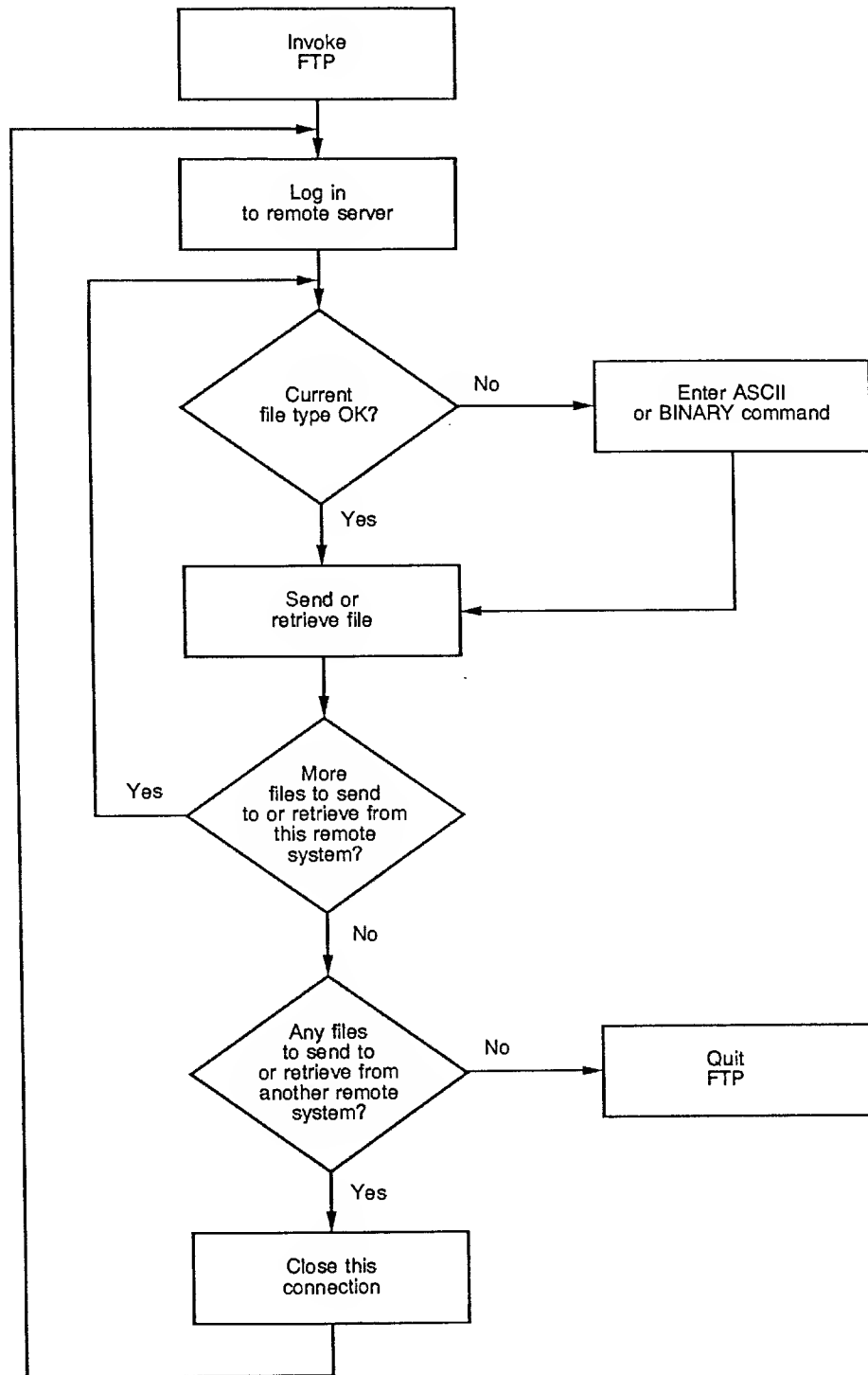
All FTP commands that you enter after you invoke FTP begin with a command code followed by an argument field. You can enter a command in uppercase or lowercase. You can represent the GET command (described later in this chapter) in any of the following ways:

GET Get get GeT G

Any symbols representing the argument field can also be entered in either uppercase or lowercase, such as A or a for ASCII TYPE. One or more spaces separate the command codes and the argument fields.

You can abbreviate all FTP commands by typing only as many letters as are required to make the abbreviation unique. For example, the RETRIEVE command (described in Chapter 4, A PRIMOS FTP Commands Reference) can be abbreviated as RET. If you enter RE, which also represents the first two letters of the RENAME command, FTP displays an error message informing you that your command is ambiguous. Many FTP commands, such as the GET command, can be represented by a single letter. After you invoke FTP, you enter HELP to display a listing of all FTP commands and their abbreviations.

The argument field of an FTP command consists of a variable-length character string that ends with a . FTP does not take any action until you enter a .



Q10155-3LA-3-2

Figure 3-1
FTP Session

Invoking FTP

Use the FTP command to invoke FTP and to open a connection to a remote host. The FTP command has the following format:

```
FTP [ { sysname }
      address }
```

where *sysname* is the name of the remote system as it appears in the HOSTS.TXT file, and *address* is the Internet address of the remote host in dot notation (for example, 123.45.6.7).

When configuring PRIMOS TCP/IP, the Network Administrator enters a list of remote system names into a configuration file named HOSTS.TXT. The Administrator may also specify the names of other remote hosts that contain this information in their configuration files. If the name that you enter with the FTP command is not in the HOSTS.TXT file, a program called the Hostname Service attempts to find the name in the configuration file of a remote host. Therefore, you may be able to enter a name that is not in the HOSTS.TXT file. Check with your Network Administrator, who can provide you with a list of remote system names that are known to PRIMOS TCP/IP.

You can also enter an Internet address that is not in the HOSTS.TXT file if the remote host identified by the number can be reached from the network to which your system is attached.

If you enter the FTP command without any arguments the ftp> prompt appears, as in the following example:

```
OK, FTP
```

```
[FTP Rev 22.0 Copyright (c) 1988, Prime Computer, Inc.]
```

```
ftp>
```

When the ftp> prompt appears, you can open a connection to a remote system, as described in the next section.

Note

When you invoke FTP, the PRIMOS command-line editor, EDIT_CMD_LINE (ECL) is enabled. ECL allows you to edit, save, and redisplay most FTP commands and arguments that you enter. ECL is not enabled for commands that prompt you to enter sensitive information such as account numbers, passwords, or project IDs.

If you have customized your environment with ECL, that environment is preserved. If you have disabled ECL with the ECL -OFF command, FTP enables the ECL default characteristics. For more information on ECL, see the *PRIMOS Commands Reference Guide*.

Connecting to a Remote Server

If you enter the FTP command with a system name, FTP displays the following prompts:

```
OK, FTP SYS88
```

```
[FTP Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
```

```
FTP is connecting to host: SYS88
```

```
220 FTP server ready.
```

```
Remote host connected.
```

```
Username (sys88:fred): freddie
```

```
331 Enter PASS command.
```

```
PASSWORD (sys88:freddie):
```

```
230 User FREDDIE logged in.
```

```
ftp>
```

If you enter the FTP command with an Internet address, FTP displays the following prompts:

```
OK, FTP 131.102.2.1
```

```
[FTP Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
```

```
FTP is connecting to remote host: 131.102.2.1
```

```
220 Prime FTP Server.
```

```
Remote host connected.
```

```
Username (131.102.2.1:mac): BMAC
```

```
331 Enter PASS Command.
```

```
Password (131.102.2.1:bmac):
```

```
230 User BMAC logged in.
```

```
ftp>
```

When you specify the remote host name or an Internet address on the command line, PRIMOS FTP attempts to open a connection to the remote FTP server. In these examples, the remote system responds with a numbered message. All numbered messages come from the remote FTP server; different FTP servers generate different messages. PRIMOS FTP generates the Remote host connected message and all other messages that are not numbered. Refer to Appendix B for a list of FTP messages.

In the above examples, FTP displays the name of the remote system and your user ID on the local system and prompts you for your remote user ID. Press to accept the default (your user ID on the local system) as your login ID on the remote system. If your remote user ID is different from your local user ID, specify another user ID and press .

FTP then prompts you for a password or project ID (if required). FTP again displays the remote host name and the user ID that you entered. When you enter a password or project ID, FTP does not echo it on the screen. PRIMOS FTP presents the user ID and password (or project ID) to the remote system.

If you enter a valid user ID, password, and/or project ID (if needed), the remote FTP server sends a numbered message and PRIMOS FTP responds with the `ftp>` prompt. You then can enter other PRIMOS FTP commands to send or to retrieve a file. If you enter a user ID, but omit a required password, you are prompted for the password.

Alternative Method of Connecting to a Remote Server

If you enter the FTP command without a system name or an Internet address, use the `OPEN` command to connect to the remote FTP server after the `ftp>` prompt appears.

The `OPEN` command has the following format:

`OPEN { sysname }
 { address }`

where *sysname* is the remote system name and *address* is the system's Internet address (for example, 123.42.0.1).

The following example illustrates the use of the `OPEN` command after you invoke FTP:

```
OK, FTP

[FTP Rev 22.0 Copyright (c) 1988, Prime Computer, Inc.]

ftp> OPEN SYS88

220 FTP server ready.
Remote host connected.
Username (sys88:fred):  Return
331 Enter PASS command.
PASSWORD (sys88:fred):
230 User FRED logged in.

ftp>
```

When you specify the remote host name, FTP attempts to establish a connection to the remote system.

You can establish a connection to only one host at a time. You must close a connection with one host before you can open a connection to another host. See the section, *Quitting PRIMOS FTP*, later in this chapter for a description of the `CLOSE` command.

If You Enter an Invalid Password

If you enter an invalid user ID when prompted or you fail to connect to a remote system, use either the `LOG` command or the `USER` command to identify yourself to the remote FTP server. The `LOG` and `USER` commands admit you to the remote system. The `LOG` command is like the

PRIMOS LOGIN command; it allows you to enter a user ID, password, and project ID as arguments on the same command line.

The LOG command has the following format:

LOG *username* [*password*] [*projectname*]

where *username* is a user ID and *projectname* is a project ID. If you omit a password or project ID, FTP prompts you to enter them (if required).

The USER command has the following format:

USER *username*

where *username* is a user ID. If the remote system requires either a password or an account number or both, you are prompted for them.

The following example shows how to use the LOG command when you enter an invalid password.

OK, **FTP**

[FTP Rev 22.0 Copyright (c) 1988, Prime Computer, Inc.]

```
ftp> OPEN SYS88
220 FTP server ready.
Remote host connected.
Username (sys88:sarah): COURTNEY
331 Enter PASS Command.
PASSWORD (sys88:courtney):
530 Login attempt failed, try again.
```

```
ftp> LOG COURTNEY TOASTER
331 Enter PASS Command.
230 User COURTNEY logged in.
```

```
ftp>
```

In this example, user SARAH's local and remote user IDs are different. Instead of accepting the local ID as her default ID, SARAH enters another user ID.

In the following example, user COURTNEY invokes the USER command to log in to the remote system after a login attempt fails. The user can only enter her user ID on the command line and must enter her password when prompted. The password is not echoed on the screen.


```
ftp> OPEN SYS88

220 FTP server ready.
Remote host connected.
Username (sys88:sarah): COURTNEY
PASSWORD (sys88:courtney):
530 Login attempt failed, try again.

ftp> USER COURTNEY
331 Enter PASS Command.
Password (sys88:courtney):
230 User Logged in.

ftp>
```

Note

When you open a connection and you are prompted to enter a user ID, password, or project ID, ECL is not enabled. If you enter the LOG command with a user ID, password, or project ID or the USER command with a user ID, ECL is enabled.

If the Connection or Login Fails

This section describes some of the error messages that FTP displays if it encounters a problem.

If the TCP/IP managing process (TCP/IP_MANAGER) is not running, FTP displays the following messages after you enter the OPEN command:

```
Cannot open connections to any remote host.
The TCP/IP product is not running on your local system.
Contact your system administrator to get TCP/IP started.
OK,
```

If FTP fails to connect you to the remote FTP server because the remote system is down or the remote FTP server is not running, FTP displays the following message:

```
Unable to connect to remote host.
ftp>
```

If the connection to the remote FTP server fails after a file transfer has started, FTP displays the following message:

```
Data connection failure: controller interface error.
ftp>
```

If one of these messages appears, exit FTP with the BYE or QUIT command and try connecting to the remote system later.

If you enter an invalid remote system name, FTP displays the following message:

```
Unknown host name
```

If you enter an Internet address with either the FTP command or the OPEN command and FTP cannot open a connection to the remote host that you specify, FTP displays the following message:

```
Network unreachable
```

If you enter an invalid password or an invalid user ID or both, the remote server and PRIMOS FTP display a message similar to the following message:

```
530 Login failed.
```

The remote server generates the number (530) that precedes the Login failed message. Different remote servers can display different messages. If a message like the above appears, find out the correct password or user ID and enter the LOG command again to log in to the remote FTP server.

Selecting File Types

At PRIMOS TCP/IP Release 2.0, PRIMOS FTP distinguishes between two types of files:

- ASCII — Intended primarily for the transfer of text files in 8-bit bytes. Files received by the 50 Series system are stored in the ASCII character set used by Prime (as described in Appendix C of the *PRIMOS User's Guide*). FTP (on the 50 Series system) inverts the high bit on incoming characters to convert standard ASCII to Prime ASCII; it also inverts the high bit on outgoing characters to convert Prime ASCII to standard ASCII. FTP also converts a line-feed (LF) character to carriage return (CR) and line-feed characters. The receiving remote system converts the 8-bit representation to its own internal form. ASCII is the default type.
- Binary — Intended for storing and retrieving binary data. FTP transfers binary data unchanged.

Notes

At Release 2.0, FTP supports only ASCII and binary types. EBCDIC and local file types are not supported.

When a binary file with an odd number of bytes is stored on a 50 Series system, PRIMOS adds a null character at the end of the file.

FTP transfers a file as a DAM file.

Changing the File Type

When you establish a connection and log in to a remote server, PRIMOS FTP establishes the default file type as ASCII. If you want to transfer binary files, you must change the default setting from ASCII to binary. Use the BINARY command or the TYPE command with the I (Image) argument, as in the following example:

```
ftp> BINARY
```

or

```
ftp> TYPE I
```

To change the file type setting back to ASCII, use the ASCII command or the TYPE command with the ASCII argument. (See Chapter 4 for a description of these commands.) Thus, to reset the file type to ASCII (the default setting) if it has been set to binary, you can use either one of the following commands:

```
ftp> ASCII
```

or

```
ftp> TYPE A
```

If you open a connection to one system, change the file type to binary, close the connection, and open a new connection to another system, PRIMOS FTP changes the file type to ASCII, the default setting.

Caution

It is the user's responsibility to check that the file type setting matches the file type. PRIMOS FTP transfers an ASCII file when the file type is set to binary or a binary file when the file type is set to ASCII. An incorrect setting can corrupt a transferred file.

Determining the Current File Type

To discover the current file type setting, enter the STATUS command in the FTP environment with no argument, as in the following example:

```
ftp> STATUS
Current transmission parameter values:
      Type: ASCII
      Mode: STREAM
```

```
Structure: FILE
Buffer Size: 1950 bytes
Transfer show flag is OFF
Local CWD: <TEST>FIRST>FTP>FRED
```

The STATUS command displays the file type, the file transfer mode, the file transfer structure, the size of the transfer buffer, and the pathname of the current working directory (CWD) on the 50 Series host. At Release 2.0, PRIMOS FTP supports only STREAM mode and FILE structure. The transfer show flag indicates whether the HASH command, which displays hash marks (#) on the screen when a file transfer is in progress, has been enabled. Refer to Chapter 4, A PRIMOS FTP Commands Reference, for a description of the MODE, STRUCTURE, and HASH commands.

Sending Files

You can send one file or multiple files to a remote system. PRIMOS FTP supports the SEND and MPUT commands. (PUT and STORE are synonyms for SEND.) You can enter a wildcard name with the MPUT command.

SEND Command

To send a file from a Prime system to a remote system, use the SEND command, which has the following format:

```
SEND local-filename [remote-filename]
```

local-filename is the name of a file in your current directory on the local 50 Series system. *remote-filename* is the name of the file after it is transferred to the remote system. The syntax of *remote-filename* must satisfy the requirements of the remote system. When you omit *remote-filename*, *local-filename* is used for the remote filename.

FTP transfers the file to the directory to which you are attached on the remote system. The section, Directory Information on the Remote System, later in this chapter describes the commands that you can use to get information about that remote directory and the files that it contains.

Note

If you send a file from a 50 Series system to a remote system and you specify the name of a file that already exists, the remote FTP server may or may not overwrite that file depending on how the remote system implements FTP. If you do not want a file overwritten, be sure you specify a unique *remote-filename*. (See the DIR command in the next chapter for instructions on obtaining a listing of remote files.)

An Example of the SEND Command: The following example shows how to use the SEND command and the messages that the remote server and FTP display.

```
ftp> SEND <EXP>PRACTICE>EXERCISE TEST
200 PORT command okay.
150 Opening data connection for exercise (192.9.201.50,5006).
226 Transfer complete.
Transferred 2056 bytes in .06 seconds (37381.61 bytes/sec, 36.51 KB/s)
ftp>
```

In this example, the remote FTP server returns numbered messages to the 50 Series user. Different remote FTP servers may generate different messages. After the last numbered message (226), PRIMOS FTP displays the file transfer rate in bytes per second (bytes/sec) and kilobytes per second (KB/s).

At the completion of this transfer, you can send another file to this remote system. If, however, you want to send a file to another remote system, you must close the session with the remote server of this system and connect to the new destination system. Use the CLOSE command (described in Chapter 4, A PRIMOS FTP Commands Reference) to close a connection without exiting FTP.

Sending a File to a Password-protected Directory

PRIMOS FTP supports sending files to password-protected directories. To send a file from a password-protected directory, you must attach to the directory in which the file resides. If you want to transfer a file that resides in a subdirectory, you must first attach to the parent directory if the parent directory is protected by a password.

When you want to send a file in a password-protected directory to a remote system, use the A command, as in either of the following examples:

```
ftp> A <TOP>TREE>BRANCH SECRET
```

or

```
ftp> A <TOP>TREE>BRANCH, SECRET
```

where <TOP>TREE>BRANCH is the full pathname of a password-protected directory and SECRET is the password. Here are two examples of using a relative pathname to attach to a password-protected directory.

```
ftp> A *>BRANCH SECRET
```

```
ftp> A *>BRANCH, SECRET
```

After you attach to a password-protected directory, you can then enter the SEND command.

If you enter an invalid password, FTP displays the message `Bad password condition encountered`. The `ftp>` prompt is displayed and your connection to the remote host remains open.

Sending Multiple Files

Use the `MPUT` command to send one or more files in your current working directory to your directory on a remote system. `MPUT` supports wildcard expansions.

Note

Before you send multiple files, check that the file types are the same (ASCII or binary) and that the file type setting matches the files that you are sending.

The `MPUT` command has the following format:


`MPUT` [*file-sys-object*] [-NQ]

where *file-sys-object* is the filename or pathname of the file or files that you want to transfer to your current directory on the remote host. You can specify a full pathname or a relative pathname. If you specify a directory name, FTP sends all the files in that directory to the remote system. If you omit *file-sys-object*, PRIMOS FTP transfers all the files in your current directory on the local system. If you specify -NQ (no query), PRIMOS FTP transfers files without prompting you to verify the filename or pathname. If you omit -NQ, FTP prompts you.

An Example of the MPUT Command: In the following example, you send all the files in your `SALES` subdirectory to the remote system:

```
ftp> MPUT *>SALES

MPUT "monday.dat" (y/n)? Y
MPUT "tuesday.dat" (y/n)? Y
MPUT "wednesday.dat" (y/n)? Y
200 PORT command okay.
150 Opening data connection for wednesday.dat (126.113.3.101,6145).
226 Transfer complete
Transferred 36288 bytes in 1.24 seconds (29193.97 bytes/sec, 28.51 KB/s).
```

When FTP prompts you, you must enter either a `Y(es)` or an `N(o)` in uppercase or lowercase. FTP does not accept `` as a default. The `MPUT` command always sends files with their names in lowercase.

If you specify the -NQ option but omit *file-sys-object*, insert a hyphen for the filename, as in the following example:

```
ftp> MPUT - -NQ
```

In the above example, FTP transfers all the files in your current directory to the remote system without asking you to verify a filename.

Wildcard Names: You can use the PRIMOS wildcard symbols @ or @@ with the MPUT command. Wildcards enable you to specify groups of files that the MPUT command can select and send. A wildcard name selects files that are either in uppercase or lowercase.

Suppose that the <EXP>TEST>PROGS directory contains the following file system objects:

CONN.COMO	NEWCL.C	NOTES	OLDPROG.C
SERVER.C	TEMP	TEST.OLD	WDDR.BIN

The following MPUT command transfers all two-component filenames that end in .C:

```
ftp> MPUT <EXP>TEST>PROGS>@.C
```

```
MPUT "newcl.c" (y/n)? Y
```

```
MPUT "oldprog.c" (y/n)? Y
```

```
MPUT "server.c" (y/n)? Y
```

If you enter the -NQ option with a wildcard, FTP transfers multiple files automatically. If FTP cannot find a filename on the remote system that matches the wildcard name, it displays the message

```
No entries selected
```

Note

Wildcards can refer only to files in one directory. For example, you cannot enter the command

```
ftp> MPUT @@>@@>@@
```

to transfer all the files in all the directories that the wildcard pathname specifies.

More Examples of the MPUT Command: The following table lists some examples of the MPUT command that you can enter.

<i>Command</i>	<i>Description</i>
MPUT	Sends all the files in your directory and asks for verification
MPUT - -NQ	Sends all the files in your directory without asking for verification
MPUT PROG.C -NQ	Sends a file named PROG.C without asking for verification
MPUT *>TEST	Sends all the files in a subdirectory named TEST and asks for verification
MPUT <SMITH>PROGS> @.BIN -NQ	Sends all the files in the <SMITH>PROGS directory that match the suffix .BIN without asking for verification

Retrieving Files

You can retrieve one file or multiple files from a remote system. PRIMOS FTP supports the GET and MGET commands. You can enter a wildcard name with the MGET command if the remote FTP server supports wildcarding.

GET Command

To retrieve a file from a remote host to the local 50 Series system, use the GET command, which has the following format:

GET *remote-filename* [*local-filename*]

remote-filename is the name of the source file on the remote system. The syntax of *remote-filename* must satisfy the requirements of the remote system. *local-filename* is the name of the file as FTP stores it in the working directory of the 50 Series system. When you omit *local-filename*, *remote-filename* is used for the the local filename. If you omit *local-filename*, *remote-filename* must be a filename, not a pathname.

Note

If *local-filename* already exists on the local host, FTP asks you whether you want to overwrite the existing file. If you answer yes, FTP creates a temporary file with a unique name and puts the contents of the retrieved file in it. FTP deletes the existing file when the transfer is completed successfully. The temporary file is renamed with the filename that you originally requested. If *local-filename* does not exist on the local host, FTP retrieves the filename with the name that you specified. If you do not want to overwrite the file, FTP does not initiate the transfer and instead displays the ftp> prompt. You can enter the GET command again with a unique filename.

If the file transfer is interrupted, FTP displays a message informing you where the partially retrieved file is stored. If the connection was not established, the empty temporary file is deleted.

An Example of the GET Command: The following example shows how to use the GET command and the messages that the remote server and FTP display after a successful transfer. In the following example, you retrieve the EXERCISE file from a remote 50 Series system.

```
ftp> GET TEST <EXP>PRACTICE>EXERCISE
200 PORT command okay.
150 Opening data connection for test (192.9.200.50,5008).
226 Transfer complete.
Transferred 256 bytes in .02 seconds (14224.01 bytes/sec, 13.89 KB/s)
ftp>
```


After the last numbered message (226), PRIMOS FTP displays the file transfer rate in bytes per second (bytes/sec) and kilobytes per second (KB/s).

If you retrieve a file from a UNIX system and enter a UNIX pathname for the filename, FTP stores the file on the 50 Series system with its pathname. Suppose that you are connected to the /usr/smith directory on a UNIX system. You enter the GET command with a UNIX pathname to retrieve a file named TESTPROG. subdirectory:

```
ftp>get /usr/smith/testprog
200 PORT command okay
150 Opening data connection for /usr/smith/testprog
(148.110.2.41,5185)
226 Transfer complete.
Transferred 3 bytes in 0.16 seconds (19.35 bytes/sec, 0.02 KB/s).
```

FTP stores the TESTPROG file as /USR/SMITH/TESTPROG.

Note

Pathnames and filenames must contain valid PRIMOS characters. You cannot retrieve files from remote systems whose operating systems support filenames that contain either a backslash (\) or a colon (:).

At the completion of a transfer, you can retrieve another file from this remote system. However, if you want to retrieve a file from another remote system, you must close the session with the remote FTP server of this system and connect to the new destination system. Use the CLOSE command (described in Chapter 4, A PRIMOS FTP Commands Reference) to close a connection without exiting FTP.

Retrieving a File From a Password-protected Directory

If you want to retrieve a file from a password-protected directory on a remote 50 Series system, you must attach to the directory where the file resides. Use the CD command to attach to the password-protected directory.

In the following example, a user is attached to the directory TREE but wants to send a file to the directory BRANCH, which is protected by the password SECRET.

```
ftp> CD TREE>BRANCH, SECRET
```

When you use the CD command, as in the above example, you must insert a comma between the pathname (TREE>BRANCH) and the password (SECRET) unless you surround the pathname and password with single or double quotation marks, as in the following example:

```
ftp> "CD TREE>BRANCH SECRET"
```

A space before or after the comma without quotation marks is not permitted.

Retrieving Multiple Files

Use the MGET command to retrieve one or more files from your current remote directory to your local directory. MGET supports wildcard expansions if the remote FTP server supports wildcarding.

Note

Before you retrieve multiple files, check that the file types are the same (ASCII or binary) and that the file type setting matches the files that you are retrieving.

The MGET command has the following format:

MGET [*remote-file-sys-object*] [-NQ]

where *remote-file-sys-object* is the filename or pathname of the file or files that you want to retrieve from the current directory of the remote host. You can enter a full pathname or a relative pathname. If you specify a directory name, FTP retrieves the files in that directory. If you omit *remote-file-sys-object*, PRIMOS FTP retrieves all the files in your current directory on the remote system. If you specify -NQ (no query), PRIMOS FTP retrieves files without prompting you to verify the filename or the pathname.

An Example of the MGET Command: The following MGET command retrieves all the files in the subdirectory LETTERS on a remote system and prompts you to verify the filenames:

```
ftp> MGET /july/letters
```

```
MGET "jones.doc" (y/n)? Y
```

```
MGET "dunn.doc" (y/n)? Y
```

```
MGET "will.doc" (y/n)? Y
```

```
200 PORT command okay.
```

```
150 Opening data connection for /bin/ls (123.123.1.103,7156) (0 bytes).
```

```
226 Transfer complete.
```

```
Transferred 126 bytes in 0.01 seconds (14007.69 bytes/sec, 13.68 KB/s).
```

```
200 PORT command okay.
```

```
150 Opening data connection for jones.doc (123.123.1.103,7157) (70 bytes).
```

```
226 Transfer complete.
```

```
Transferred 1160 bytes in 0.03 seconds (37421.49 bytes/sec, 36.54 KB/s).
```

When FTP prompts you, you must respond Y(ES) or N(O) in uppercase or lowercase. FTP does not accept as a default.

In the above example, /bin/ls refers to a temporary list of files that FTP retrieves. When you retrieve multiple files, the remote FTP server lists only the first file on the list (jones.doc) before it displays the Transfer complete message.

If you enter Y and the file exists locally, FTP asks you whether you want to overwrite it. FTP displays this overwrite query only if you do not specify the -NQ option. If any file to be retrieved is on the local system and you use -NQ, FTP overwrites it without notice. If you enter the -NQ option, FTP retrieves multiple files automatically.

To retrieve all the files from a remote directory without FTP prompting you, specify the -NQ option, omit *remote-file-sys-object*, and insert a hyphen for the filename, as in the following example:

```
ftp> MGET - -NQ
```

The above command retrieves all the files from your current working directory on the remote host to the 50 Series system without prompting you to verify filenames.

Wildcard Names: The wildcard symbols that you enter must conform to the wildcarding conventions that the remote host supports (for example, * for UNIX, or @ for PRIMOS systems). The wildcard may retrieve files with names in either uppercase or lowercase, depending on the implementation of wildcarding on the remote system.

Suppose your working directory on a remote UNIX system contains the following file system objects (as displayed by the LS command):

```
conn.old
newcl.c
notes.doc
oldprog.c
server.c
temp
test.old
wddr.bin
```

The following MGET command retrieves all two-component filenames that end in .C:

```
ftp> MGET *.c
```

```
MGET newcl.c (y/n)? Y
MGET oldprog.c (y/n)? Y
MGET server.c (y/n)? Y
```

You must be careful to specify the filename in uppercase or lowercase if you are retrieving files from a UNIX system. In the above example, the wildcard name (*.c) is in lowercase because the FTP server on the remote system is case-sensitive.

If you enter the -NQ option with a wildcard, FTP transfers multiple files automatically.

Note

A wildcard name can refer only to files in one remote directory. For example, you cannot enter the command

```
ftp> MGET *. */*. */*. *
```

to retrieve all the files in all the directories that the wildcard pathname specifies.

More Examples of the MGET Command: The following table lists some examples of the MGET command that you can enter.

<i>Command</i>	<i>Description</i>
MGET	Retrieves all the files in your remote directory and asks for verification
MGET - -NQ	Retrieves all the files in your remote directory without asking for verification
MGET /etc/host -NQ	Retrieves the /etc/host file from a remote system without asking for verification
MGET /test	Retrieves all the files in a remote subdirectory named /test and asks for verification
MGET /smith/ progs/*.bin -NQ	Retrieves all the files in the /smith/progs directory that match the suffix .bin without asking for verification

Executing Other Commands From Within FTP

After you invoke PRIMOS FTP, you can execute some commands that are comparable to PRIMOS commands. The next two sections describe these commands.

Directory Information on the 50 Series Host

The following commands enable you to obtain information about files and directories on the 50 Series host or perform other tasks:

! PRIMOS_command

Enables you to execute any PRIMOS command subject to verification. Precede the PRIMOS command that you specify with an exclamation mark (!) and a space. If the command is more than one word, surround the command with single or double quotation marks. FTP submits the command as a string to the local host without checking the syntax. Therefore, you must be certain that the syntax of *PRIMOS_command* is correct. If you enter ! and omit *PRIMOS_command*, FTP prompts you to enter one.

?

Displays information about FTP commands. A synonym for HELP.

A *local-directory*

Changes your current working directory to *local-directory*.

CHDIR *local-directory*

Changes your current working directory to *local-directory*. A synonym for the A command.

HASH

Turns on or off the displaying of hash marks (#) on the screen while a file transfer is in progress.

LCD *local-directory*

Changes your current working directory to *local-directory*. A synonym for the A command.

LD

Lists the files and subdirectories in your current local PRIMOS working directory.

SHOW

Turns on or off the displaying of hash marks (#) on the screen while a file transfer is in progress. A synonym for HASH.

STATUS

Displays information about the current local working directory and the transfer parameters of type, mode, and structure.

Chapter 4, A PRIMOS FTP Commands Reference, provides more information on these commands.

Examples of the ! Command

You can execute PRIMOS commands from within FTP. The commands that you enter are invoked on the local 50 Series system. PRIMOS validates all commands that you submit. See the *PRIMOS Commands Reference Guide* for a description of PRIMOS commands.

Precede the PRIMOS command that you enter with an exclamation mark (!), as in the following example:

```
ftp> ! TIME
```

```
Time used: 00h 04m connect, 00m 24s CPU, 00m 12s I/O.
```

If you enter ! without an argument, FTP prompts you for a command, as in the following example:

```
ftp> !
PRIMOS command: TIME
Time used: 00h 04m connect, 00m 24s CPU, 00m 12s I/O.
```

You must surround the PRIMOS command with single or double quotation marks if the PRIMOS command is more than one word, as shown in the following examples:

```
ftp> ! 'STATUS COMM'
```

Controller	Type	Device	Total-Lines		Bad-Lines	
		Address	Async	Sync	Async	Sync
LHC300		56	No Information		No Information	
AMLC	DMQ	54	16	0	No Information	

```
ftp> !
PRIMOS command: "STATUS COMM"
```

Controller	Type	Device	Total-Lines		Bad-Lines	
		Address	Async	Sync	Async	Sync
LHC300		56	No Information		No Information	
AMLC	DMQ	54	16	0	No Information	

Error Messages: If you enter a PRIMOS command without entering the ! before the command, FTP displays the following message:

```
ftp> TIME
"TIME" Unrecognized command.
```

If you omit the quotation marks from a command that is longer than one word, FTP displays the following message:

```
ftp> ! STATUS COMM
Multiple arguments should be quoted.
```

PRIMOS verifies all commands that you submit. If you enter an invalid PRIMOS command, FTP displays the appropriate error message, as in the following examples:

```
ftp> ! "COPY FILE.BIN <EXP>PROGS>=="
```

```
Slave validation error. Unable to attach "<EXP>PROGS" (copy)
```

```
ftp> ! TIMER
```

```
Not found. TIMER (std$cp)
```

Directory Information on the Remote System

The following commands enable you to get information about files and directories on the remote system. Chapter 4 provides complete information on these commands.

CD *remote-directory*

Changes the working directory on the remote system to *remote-directory*.

CWD *remote-directory*

Changes the working directory on the remote system to *remote-directory*. A synonym for the CD command.

DIR [*remote-directory*] [*local-filename*]

Displays detailed information about the contents of *remote-directory* on the screen or puts the contents of *remote-directory* in *local-filename*.

LIST [*remote-directory*] [*local-filename*]

Displays detailed information about the contents of *remote-directory* on the screen or puts the contents of *remote-directory* in *local-filename*. A synonym for the DIR command.

LS [*remote-directory*] [*local-filename*]

Displays a list of the filenames and subdirectories in *remote-directory* or puts a list of the files in *remote-directory* in *local-filename*.

NLST [*remote-directory*] [*local-filename*]

Displays a list of the filenames and subdirectories in *remote-directory* or puts a list of the files in *remote-directory* in *local-filename*. A synonym for the LS command.

PWD

Displays the name of the current working directory on the remote system.

FTP Help Facility

To obtain a brief description of any FTP command, use the following command after you have invoked FTP:

HELP *command*

If you omit *command* (a specific PRIMOS FTP command) a listing of PRIMOS FTP commands is displayed and you are instructed what to enter in order to display more information on each command.

Quitting PRIMOS FTP

If you are connected to one remote host but want to transfer a file to or from another remote host, you must close the connection to the first host. To break a connection with a remote system but remain within FTP, enter the `CLOSE` command:

```
ftp> CLOSE
221 Goodbye
Remote host disconnected.
```

```
ftp>
```

After you close a connection, you can issue the `OPEN` command to establish a connection to another host.

To end an FTP session and return to PRIMOS command level, enter the `QUIT` or `BYE` command:

```
ftp> QUIT
221 Goodbye.
Remote host disconnected.
Exiting due to quit request
```

```
Exiting FTP
OK,
```

Summary of FTP Commands

This section briefly describes FTP commands. See Chapter 4, A PRIMOS FTP Commands Reference, for a complete description of these commands.

FTP Commands

Table 3-2 summarizes the FTP commands that a user on a 50 Series system can invoke at Release 2.0. These commands run the FTP program. Table 3-3 lists all the rights that you need to invoke specific FTP Commands.

Table 3-2
FTP User Commands

<i>Command</i>	<i>Description</i>
✓ ACCOUNT	Allows you to enter an account number when a remote system requires an account number to log in.
✓ ASCII	Sets the file transfer type to ASCII.
✓ BINARY	Sets the file transfer type to binary so that binary image files can be transferred. A synonym for IMAGE.
✓ BYE	Terminates the FTP session with the remote server and exits FTP. A synonym for QUIT.
✓ CD	Changes the working directory on the remote system to a remote directory that you specify.
✓ CLOSE	Terminates the FTP session with the remote server, but enables you to stay within FTP.
CONTROL-P	Terminates an FTP session with the remote server, exits FTP, and returns to PRIMOS command level. CONTROL-P is equivalent to QUIT.
CWD	Changes the working directory on the remote system to a remote directory that you specify. A synonym for CD.
✓ DELETE	Deletes a remote file that you specify from the working directory on the remote system.
✓ DIR	Displays or saves the contents of a remote directory in a file.
✓ GET	Retrieves a remote file that you specify and stores it on the local system. A synonym for RECV and RETR.
IMAGE	Sets the file transfer type to binary so that binary image files can be transferred. A synonym for BINARY.
LIST	Displays or saves the contents of a remote directory in a file. A synonym for DIR.
LOG	Identifies the user to the remote FTP server.
✓ LS	Displays or saves in a file a list of filenames and subdirectories in the remote directory.
✓ MGET	Retrieves multiple files from a remote system to a local system.
✓ MKDIR	Creates a directory on the remote system.
✓ MODE	Sets the file transfer mode to STREAM mode. STREAM mode is the default mode and is the only mode currently supported.
✓ MPUT	Sends multiple files from a local system to a remote system.

Table 3-2
FTP User Commands – Continued

<i>Command</i>	<i>Description</i>
NLST	Displays or saves in a file a list of filenames and subdirectories in the remote directory. A synonym for LS.
✓ OPEN	Establishes a connection to an FTP server on a remote system.
✓ PUT	Transfers a file from a local system to a remote system and stores it on the remote system. A synonym for SEND and STORE.
✓ PWD	Displays the name of the remote system's current working directory.
✓ QUIT	Terminates the FTP session with the remote server and exits FTP. A synonym for BYE.
✓ RECV	Retrieves a remote file that you specify and stores it on the local system. A synonym for GET and RETRIEVE.
✓ RENAME	Renames a file on the remote system.
RETRIEVE	Retrieves a remote file that you specify and stores it on the local system. A synonym for GET and RECV.
✓ RMDIR	Deletes an empty directory on the remote system.
✓ SEND	Transfers a file from a local system to a remote system and stores it on the remote system. A synonym for PUT and STORE.
STORE	Transfers a file from a local system to a remote system and stores it on the remote system. A synonym for PUT and SEND.
STRUCTURE { <i>struct in UNIX</i> }	Sets the file transfer structure. File structure is the only structure that the current version of FTP supports. File structure is the default.
✓ TYPE	Sets the file-transfer type. The default type is ASCII. ASCII and binary are the only types supported in the current version.
✓ USER	Identifies the user to the remote FTP server.

Table 3-3
Access Rights for User FTP Commands

<i>Command</i>	<i>Local User Access Rights</i>
CD, CWD	None
DELETE	None
GET, RETRIEVE, RECV	ALU (if file does not exist) ADLUR (if file exists)
LS, NLST	None (if list displayed on screen)
LIST, DIR	ALU (if list written to new file) ADLUR (if list overwrites existing file)
MKDIR	None
MGET	ADLUR
MPUT	ADLUR
PWD	None
RENAME	None
RMDIR	None
SEND, STORE, PUT	UR

Sample Session

The following example shows user BERNIE performing the steps listed below:

- Invoking FTP on a 50 Series system
- Opening a connection to a remote system with an Internet address of 193.44.5.2
- Logging in to the remote system with the USER command
- Displaying the name of the current working directory on the remote system with the PWD command
- Creating a subdirectory with the MKDIR command
- Changing directories with the CD command
- Setting the FILE type to binary with the BIN command
- Using the HASH command to enable the display of hash marks during the file transfer
- Checking the new settings with the STATUS command

- Transferring a binary file with the SEND command
- Using the DIR command to list the transferred file in the subdirectory
- Using the QUIT command to end the FTP session

PRIMOS FTP generates unnumbered messages. The remote system's FTP server generates numbered messages, as shown in the example.

OK, *FTP*

[FTP Rev 22.0 Copyright (c) 1988, Prime Computer, Inc.]

```
ftp> OPEN 190.44.5.2
220 FTP server (Version 4.20 Wed Sep 29 19:56:27 PST 1988) ready.
Remote host connected.
Username (193.44.5.2:ernie):bernie
331 Enter PASS command.
PASSWORD (193.44.5.2:bernie):
230 User BERNIE logged in.
```

```
ftp> PWD
251 "XLN" is current directory.
```

```
ftp> MKDIR GREEN
200 MKDIR command okay.
```

```
ftp> CD GREEN
200 CWD command okay.
```

```
ftp> BIN
200 Type set to I.
```

```
ftp> HASH
SHOW/HASH is turned ON
```

```
ftp> STATUS
```

```
Current transmission parameter values:
      Type: IMAGE
      Mode: STREAM
      Structure: FILE
      Buffer size: 1950 bytes
      Transfer show flag is ON
      Local CWD: <TPUBS>SAMP
```

```
ftp> SEND TEST.BIN
200 PORT command okay.
150 Opening data connection for test.bin (190.44.5.2,5010).
#####
#####
#####
#####
Transferred 8020 bytes in .40 seconds (37381.61 bytes/sec, 36.41 KB/s)
226 Transfer complete.

ftp> DIR
200 PORT command okay.
200 Type set to A.
150 Opening data connection for (190.44.5.2,5011).
TEST      BIN      312578   9-01-88   4:08p
226 Transfer complete.
200 Type set to I.

ftp> QUIT
221 Goodbye.
Remote host disconnected.
Exiting due to quit request.
OK,
```


A PRIMOS FTP Commands Reference

This chapter describes PRIMOS FTP commands for a user on a 50 Series host. PRIMOS FTP commands can be classified into two categories:

- Local commands, similar to PRIMOS commands, that you can execute after you invoke FTP on the local 50 Series system. Local commands work on the files or directories on your local Prime host on which you log in.
- PRIMOS FTP commands that are sent to the FTP server on the remote system. FTP commands are sent to the remote system from the local 50 Series system and affect files or directories on the remote machine.

A user on the 50 Series system can use FTP commands to send files to and retrieve files from the remote system.

Note

In all PRIMOS FTP commands, *local-file* or *local-directory* is the name of a file or directory on the local 50 Series system. *remote-file* or *remote-directory* is the name of the file or directory on the remote host or workstation. When you use the name of a file or a directory in a command that you issue from the 50 Series system, the syntax of that name must satisfy the syntax requirements of the system where the file or directory resides.

Format of FTP Commands

All PRIMOS FTP commands begin with a command code followed by an argument field. You can enter a command in uppercase or lowercase. You can enter the GET command (described later in this chapter) in any of the following ways:

GET Get get GeT G

Any symbols representing the argument field can also be entered either in uppercase or lowercase, such as A or a for ASCII TYPE. One or more spaces separate the command codes and the argument fields.

You can abbreviate all PRIMOS FTP commands by typing only as many letters as are required to make the abbreviation unique. For example, the RETRIEVE command (described later in this chapter) can be abbreviated as RET. If you enter RE, which also represents the first two letters of the RENAME command, PRIMOS FTP displays an error message informing you that your command is ambiguous. Many PRIMOS FTP commands, such as the GET command, can be represented by a single letter. After you invoke FTP, enter HELP to display a listing of all FTP commands and their abbreviations.

The argument field of a PRIMOS FTP command consists of a variable-length character string that ends with . PRIMOS FTP does not take any action until you enter .

Local Commands

The following commands are equivalent to some PRIMOS commands. They enable you to perform tasks on your local system from within PRIMOS FTP. The commands are listed in alphabetical order.

► A *local-directory* [*password*]

Changes your current directory to another directory. *local-directory* is the pathname of the directory that becomes the new current directory in the 50 Series system. The local working directory on the 50 Series system is changed to *local-directory*. *password* is the password if the local directory is protected by a password.

Specify pathnames as you would with the PRIMOS ATTACH command. Use the directory name to attach to a top-level directory, as in the following example:

```
ftp> A FOREST
```

Begin the pathname with an asterisk (*) to attach to a directory subordinate to your own, as in the following example:

```
ftp> A *>BRANCH
```

If you want to attach to a password-protected directory, use the A command, as in either of the following examples:

```
ftp> A *>BRANCH SECRET
```

or

```
ftp> A *>BRANCH, SECRET
```

where *>BRANCH is the relative pathname of a password-protected directory and SECRET is the password.

If you enter an invalid password, FTP displays the message *Bad password condition encountered*. The `ftp>` prompt is displayed and your connection to the remote host remains open.

The `A` command is a synonym for the `CHDIR` and `LCD` commands.

► ***CHDIR local-directory [password]***

Changes your current directory to another directory. *local-directory* is the full pathname of the directory that becomes the new current directory in the 50 Series system. The local working directory on the 50 Series system is changed to *local-directory*. *password* is the password if the local directory is protected by a password.

Specify pathnames as you would with the PRIMOS `ATTACH` command. Use the directory name to attach to a top-level directory, as in the following example:

```
ftp> CHDIR FOREST
```

Begin the pathname with an asterisk (*) to attach to a directory subordinate to your own, as in the following example:

```
ftp> CHDIR *>BRANCH
```

If you want to attach to a password-protected directory, use the `CHDIR` command, as in either of the following examples:

```
ftp> CHDIR *>BRANCH SECRET
```

or

```
ftp> CHDIR *>BRANCH, SECRET
```

where `*>BRANCH` is the relative pathname of a password-protected directory and `SECRET` is the password.

If you enter an invalid password, FTP displays the message *Bad password condition encountered*. The `ftp>` prompt is displayed and your connection to the remote host remains open.

The `CHDIR` command is a synonym for the `A` and `LCD` commands.

► ***! PRIMOS_command***

Enables you to execute any PRIMOS command subject to verification. Precede the PRIMOS command that you specify with an exclamation mark (!) and a space. If the command is more than one word, surround the command with single or double quotation marks. FTP submits the command as a string to the local host without checking the syntax. Therefore, you must be

certain that the syntax of *PRIMOS_command* is correct. If you enter ! and omit *PRIMOS_command*, FTP prompts you to enter a command. The PRIMOS command that you enter cannot be an abbreviation.

► HASH

Turns on or off the displaying of hash marks (#) on the screen while your file transfer is in progress. When HASH is enabled, one hash mark is displayed for every FTP buffer of data transferred or received. The initial setting for the HASH command when you log in is OFF. The STATUS command, described below, indicates the size of the FTP buffer and whether the HASH command is set to ON or OFF.

The HASH command is a synonym for the SHOW command.

► A *local-directory* [*password*]

Displays information about *command*. HELP also displays the syntax of each command. If you enter HELP and omit the name of the command, PRIMOS FTP displays a list of all commands that it supports.

► LCD *local-directory* [*password*]

Changes your current directory to another directory. *local-directory* is the full pathname of the directory that becomes the new current directory in the 50 Series system. The local working directory on the 50 Series system is changed to *local-directory*. *password* is a password if the local directory is protected by a password.

Specify pathnames as you would with the PRIMOS ATTACH command. Use the directory name to attach to a top-level directory, as in the following example:

```
ftp> LCD FOREST
```

Begin the pathname with an asterisk (*) to attach to a directory subordinate to your own, as in the following example:

```
ftp> LCD *>BRANCH
```

If you want to attach to a password-protected directory, use the LCD command, as in either of the following examples:

```
ftp> LCD *>BRANCH SECRET
```

or

```
ftp> LCD *>BRANCH, SECRET
```

where *>BRANCH is the relative pathname of a password-protected directory and SECRET is the password.

If you enter an invalid password, FTP displays the message *Bad password condition encountered*. The `ftp>` prompt is displayed and your connection to the remote host remains open.

The `LCD` command is a synonym for the `A` and `CHDIR` commands.

► **LD** [*pathname*] [*wild1...wild15*] [*options*]

Lists the contents of a directory and, optionally, the attributes of the entries in the current local 50 Series system working directory.

pathname identifies the directory to be listed and contains the first wildcard name. The default *pathname* is your current directory and the default wildcard name is `@`.

wild1...wild15 means that the default `@` wildcard name can be replaced with a total of 15 wildcard names.

The following command lists entries in the current directory that begin with `SALE` or end with `.MEMO`:

```
ftp> LD SALE@ @.MEMO
```

See the *PRIMOS Commands Reference Guide* for more information on the `LD` command.

► **? command**

Displays an informative message about *command*. If you enter `?` and omit the name of the command, PRIMOS FTP displays a list of all commands that it supports.

The `?` command is a synonym for the `HELP` command.

► **SHOW**

Turns on or off the displaying of hash marks (#) on the screen while your file transfer is in progress. When `SHOW` is enabled, one hash mark is displayed for every FTP buffer of data transferred or received. The initial setting for the `SHOW` command when you log in is `OFF`. The `STATUS` command, described below, indicates the size of the FTP buffer and whether the `SHOW` command is set to `ON` or `OFF`.

The `SHOW` command is a synonym for the `HASH` command.

► **STATUS**

Displays current information about the settings of the `MODE`, `STRUCTURE`, `TYPE`, and `HASH` commands. For example,

```
ftp> STATUS
Current transmission parameter values:
      Type: ASCII
      Mode: STREAM
      Structure: FILE
      Buffer Size: 1950 bytes
      Transfer show flag is OFF
      Local CWD: <FOREST>BRANCH>LEAF
```

The MODE, STRUCTURE, and TYPE commands are described in the next section. The transfer show flag indicates whether the HASH command has been toggled on or off. Buffer size is the size of an FTP buffer indicated by one hash mark (#). CWD is the current working directory on the 50 Series system.

PRIMOS FTP User Commands

Users on a local 50 Series system host can invoke the following PRIMOS FTP commands. These commands send messages to the FTP program and manage files and directories on the remote system. The commands are listed in alphabetical order.

► ACCOUNT *account-number*

Allows you to enter *account-number* or another identifier when a remote system requires an account number or identifier to log in. If you enter the ACCOUNT command and omit *account-number*, PRIMOS FTP prompts you to enter a number.

If the remote system is a 50 Series system and you are required to enter a project ID to log in, the ACCOUNT command prompts you to enter a project ID.

► ASCII

Sets the file transfer representation type to ASCII. Use the ASCII type to transfer text files. ASCII is the default type. PRIMOS FTP inverts the high bit on incoming characters to convert standard ASCII to Prime ASCII; it also inverts the high bit on outgoing characters to convert Prime ASCII to standard ASCII. FTP also converts a line-feed (LF) character to carriage return (CR) and line-feed characters.

► BINARY

Sets the file transfer representation type to binary. The default setting for PRIMOS FTP is ASCII. If you want to transfer binary files, use the BINARY command. You can also use the TYPE command, described below.

The BINARY command is a synonym for the IMAGE command.

► BYE

Terminates a PRIMOS FTP session with the remote server and exits PRIMOS FTP.

The BYE command is a synonym for the QUIT command.

► CD *remote-directory* [*password*]

Changes the current working directory on the remote system to *remote-directory*. It does not alter login or transfer parameters that you may have entered before you invoked the CD command.

remote-directory is a pathname specifying a directory on the remote system. If the remote system is a 50 Series system, begin the pathname with an asterisk (*) to attach to a directory subordinate to your current directory.

password is a password if the remote directory on a 50 Series system is protected by a password.

When you want to attach to a remote directory on a 50 Series system, use the CD command, as in either of the following examples:

```
ftp> CD "*>BRANCH SECRET"
```

or

```
ftp> CD *>BRANCH, SECRET
```

where *>BRANCH is the relative pathname of a password-protected directory and SECRET is the password. If you omit the comma between the pathname and the password, you must enclose the pathname and password in double or single quotation marks.

If you enter an invalid password, FTP displays the message Bad password condition encountered. The ftp> prompt is displayed and your connection to the remote host remains open.

The CD command is a synonym for the CWD command.

► CLOSE

Terminates a PRIMOS FTP session with the remote server. When you issue the CLOSE command, control is returned to PRIMOS FTP and you remain in PRIMOS FTP. Use the CLOSE command to end a session with a remote system for the purpose of opening a connection to another remote system.

► CONTROL-P

Suspends a PRIMOS FTP session with the remote server. You are given the option of exiting PRIMOS FTP, continuing the current session, or aborting the current FTP session. When you

enter CONTROL-P, files that are currently open may remain open. If you enter CONTROL-P while a file transfer is in progress, the file transfer may be disrupted.

The CONTROL-P character is equivalent to the QUIT command if you exit the current session. CONTROL-P is equivalent to the CLOSE command if you abort the current session.

Note

The only recommended use of CONTROL-P is to suspend a file transfer that is in progress.

► CWD *remote-directory*

Changes the current working directory on the remote system to *remote-directory*. It does not alter login or transfer parameters that you might have entered before you invoked the CWD command.

remote-directory is a pathname specifying a directory on the remote system. If the remote system is a 50 Series system, begin the pathname with an asterisk (*) to attach to a directory subordinate to your current directory.

When you want to attach to a remote directory on a 50 Series system, use the CWD command, as in either of the following examples:

```
ftp> CWD "*>BRANCH SECRET"
```

or

```
ftp> CWD *>BRANCH, SECRET
```

where **>BRANCH* is the relative pathname of a password-protected directory and *SECRET* is the password. If you omit the comma between the pathname and the password, you must enclose the pathname and password in double or single quotation marks.

If you enter an invalid password, FTP displays the message *Bad password condition encountered*. The `ftp>` prompt is displayed and your connection to the remote host remains open.

The CWD command is a synonym for the CD command.

► DELETE *remote-file*

The DELETE command deletes *remote-file* from the working directory on the remote system.

If you are connected to a remote PRIMOS FTP server, you can use the DELETE command to delete an empty directory.

► **DIR** [*remote-directory*] [*local-filename*]

Writes a detailed listing of the contents of *remote-directory* in *local-filename*. If you omit *remote-directory*, the DIR command writes the file with the listing in your current working directory on your local 50 Series system. When you specify *local-filename* and omit *remote-directory*, insert a hyphen in the command line, as in the following example:

```
ftp> DIR - testfile
```

If you omit *local-filename*, FTP lists the filenames in *remote-directory* on the screen.

If *local-filename* already exists on the local host, FTP asks you whether you want to overwrite the existing file. If you answer yes, FTP creates a temporary file with a unique name and puts the contents of the directory in it. Once the operation is completed successfully, FTP deletes the existing file and renames the temporary file with the filename that you originally requested. If you respond no, FTP suspends the transfer and displays the ftp> prompt. You can enter the command with a unique filename.

If the operation is interrupted, FTP displays a message informing you where the partially retrieved directory is stored. If no data were transferred, the empty temporary file is deleted.

The DIR command is a synonym for the LIST command.

► **GET** *remote-filename* [*local-filename*]

Retrieves *remote-filename* from the remote system and stores it on the local 50 Series system as *local-filename*. If you omit *local-filename*, PRIMOS FTP uses *remote-filename* for the local filename. If you omit *local-filename*, *remote-filename* must be a filename, not a pathname. The syntax of *remote-filename* must satisfy the requirements of the remote system.

If *local-filename* already exists on the local host, FTP asks you whether you want to overwrite the existing file. If you answer yes, FTP creates a temporary file with a unique name and puts the contents of the retrieved file in it. Once the operation is completed successfully, FTP deletes the existing file and renames the temporary file with the filename that you originally requested. If you respond no, FTP suspends the transfer and displays the ftp> prompt. You can enter the command with a unique filename.

If the file transfer is interrupted, FTP displays a message informing you where the partially retrieved file is stored. If no data were transferred, the empty temporary file is deleted.

PRIMOS FTP uses the current setting for file type when it transfers the file. For example, if the file type is set to ASCII, FTP uses this setting.

Note

When a binary file with an odd number of bytes is stored on a 50 Series system, PRIMOS adds a null character at the end of the file. When you send the file, FTP strips off the extra null character.

The GET command is a synonym for the RECV and RETRIEVE commands.

► **IMAGE**

Sets the file transfer representation type to binary so that binary files can be transferred.

The IMAGE command is a synonym for the BINARY command.

► **LIST** [*remote-directory*] [*local-filename*]

Writes a detailed listing of the contents of *remote-directory* in *local-filename*. If you omit *remote-directory*, the LIST command writes the listing in the current working directory. When you specify *local-filename* and omit *remote-directory*, insert a hyphen in the command line, as in the following example:

```
ftp> LIST - testfile
```

If you omit *local-filename*, PRIMOS FTP lists the filenames in *remote-directory* on the screen.

If *local-filename* already exists on the local host, FTP asks you whether you want to overwrite the existing file. If you answer yes, FTP creates a temporary file with a unique name and puts the contents of the directory in it. Once the operation is completed successfully, FTP deletes the existing file and renames the temporary file with the filename that you originally requested. If you respond no, FTP suspends the transfer and displays the ftp> prompt. You can enter the command with a unique filename.

If the operation is interrupted, FTP displays a message informing you where the partially retrieved directory is stored. If no data were transferred, the empty temporary file is deleted.

The LIST command is a synonym for the DIR command.

► **LOG** *username* [*password*] [*projectname*]

Identifies you to the remote FTP server so that you can access its file system. *username* is a user ID that the remote system may require. *projectname* is a project ID that the remote system may require. You can enter your user ID, password, and project ID on the same command line.

If you omit *password* and *projectname* and the remote server requires them, you are prompted after local echo and ECL are disabled.

► **LS** [*remote-directory*] [*local-filename*]

Writes a list of the filenames and subdirectories in *remote-directory* in *local-filename*. If you omit *remote-directory*, the LS command writes the list of files in your current working directory on your local system. When you specify *local-filename* and omit *remote-directory*, insert a hyphen in the command line, as in the following example:

```
ftp> LS - testfile
```

If you omit *local-filename*, PRIMOS FTP lists the filenames in *remote-directory* on the screen.

If *local-filename* already exists on the local host, FTP asks you whether you want to overwrite the existing file. If you answer yes, FTP creates a temporary file with a unique name and puts the list of files in it. Once the operation is completed successfully, FTP deletes the existing file and renames the temporary file with the filename that you originally requested. If you respond no, FTP suspends the transfer and displays the `ftp>` prompt. You can enter the command with a unique filename.

If the operation is interrupted, FTP displays a message informing you where the partially retrieved list of files is stored. If no data were transferred, the empty temporary file is deleted.

The `LS` command is a synonym for the `NLST` command.

Note

You can enter a wildcard name with the `LS` command if the remote FTP server supports wildcard expansions. The wildcard symbols that you enter must conform to the wildcarding conventions that the remote host supports (for example, `*` for UNIX, or `@` for PRIMOS systems).

► `MGET [remote-file-sys-object] [-NQ]`

Retrieves one or more files from a remote host to your current directory on the local 50 Series system. `MGET` supports wildcard expansions if the remote server supports wildcarding.

remote-file-sys-object is the filename or pathname of the file or files that you want to retrieve from the current directory of the remote host. You can enter a full pathname or a relative pathname. If you specify a directory name, FTP retrieves the contents of that directory. If you omit *remote-file-sys-object*, PRIMOS FTP retrieves the entire contents of your current directory on the remote system.

If you specify `-NQ` (no query), PRIMOS FTP retrieves files without prompting you to verify the filename or pathname. If you specify the `-NQ` option but omit *remote-file-sys-object*, insert a hyphen for the filename, as in the following example:

```
ftp> MGET - -NQ
```

The above command retrieves all the files in your current working directory on the remote host to the 50 Series system without prompting you to verify filenames.

The wildcard symbols that you enter must conform to the wildcarding conventions that the remote host supports (for example, `*` for UNIX, or `@` for PRIMOS systems). A wildcard name can refer only to files in one remote directory. For example, you cannot enter the command

```
ftp> MGET *.* /*.*/*.*
```

to retrieve all the files in all the directories that the wildcard pathname specifies. The wildcard may retrieve files with names in either uppercase or lowercase, depending on the implementation of wildcarding on the remote system.

If you specify a filename or a pathname with a wildcard, FTP retrieves the contents of the specified local directory that match the wildcard specification. Suppose your working directory on a remote 50 Series system contains the following file system objects (as displayed by the LS command):

```
conn.old
newcl.c
notes
oldprog.c
server.c
temp
test.old
wddr.bin
```

The following MGET command transfers all two-component filenames that end in lowercase .c:

```
ftp> MGET *.c

MGET "newcl.c" (y/n)? Y
MGET "oldprog.c" (y/n)? Y
MGET "server.c" (y/n)? Y
200 PORT command okay.
150 Opening data connection for /bin/ls (123.126.1.105,8556) (0 bytes).
226 Transfer complete.
Transferred 126 bytes in 0.01 seconds (14007.69 bytes/sec, 13.68 KB/s).
200 PORT command okay.
150 Opening data connection for newcl.c (123.126.1.105,8557) (1333 bytes).
226 Transfer complete.
Transferred 1367 bytes in 0.03 seconds (50628.83 bytes/sec, 49.44 KB/s).
```

When FTP prompts you, you must respond Y(ES) or N(O) in uppercase or lowercase. FTP does not accept Return as a default.

In the above example, /bin/ls refers to a temporary list of files that FTP retrieves. When you retrieve multiple files, the remote FTP server lists only the first file (newcl.c) on the list before it displays the Transfer complete message.

When FTP prompts you, you must enter either a Y or an N in uppercase or lowercase. FTP does not accept Return as a default. If you enter the -NQ option with a wildcard, FTP transfers multiple files automatically.

If you enter Y and the file exists locally, FTP asks you whether you want to overwrite it. FTP displays this overwrite query only if you do not specify the -NQ option. If any file to be retrieved is on the local system and you use -NQ, FTP overwrites it without notice. If you enter the -NQ option, FTP retrieves multiple files automatically.

► MKDIR *remote-directory*

Creates *remote-directory* on the remote system.

► MODE [*mode-name*]

Sets the file transfer mode to *mode-name*. PRIMOS FTP supports the transfer of data as a stream of bytes (STREAM mode). STREAM mode is the default mode and is the only mode supported.

► MPUT [*file-sys-object*] [-NQ]

Sends one or more files to a remote host. MPUT supports wildcard expansions. The MPUT command always sends files with their names in lowercase.

file-sys-object is the filename or pathname of the file or files that you want to transfer to the current directory of the remote host. You can specify a full pathname or a relative pathname. If you specify a directory name, FTP sends the contents of that directory to the remote system. If you omit *file-sys-object*, PRIMOS FTP transfers the contents of your current directory on the local system.

If you specify -NQ (no query), PRIMOS FTP transfers files without prompting you to verify the filename or pathname. If you specify the -NQ option but omit *file-sys-object*, insert a hyphen for the filename, as in the following example:

```
ftp> MPUT - -NQ
```

In the above example, FTP transfers all the files in your current directory to the remote system without asking you to verify a filename.

MPUT supports the wildcard symbols @ or @@. Wildcards can only refer to files in one directory. For example, you cannot enter the command

```
ftp> MPUT @@>@@>@@
```

to transfer all the files in all the directories that the wildcard pathname specifies. The wildcard name selects files that are in either uppercase or lowercase.

If you specify a filename or a pathname with a wildcard, FTP transfers the contents of the specified local directory that match the wildcard specification. Suppose the <EXP>TEST>PROGS directory contains the following file system objects:

CONN.COMO	NEWCL.C	NOTES	OLDPROG.C
SERVER.C	TEMP	TEST.OLD	WDDR.BIN

The following MPUT command transfers all two-component filenames that end in .C:

```
ftp> MPUT <EXP>TEST>PROGS>@.C
```

```
MPUT "newcl.c" (y/n)? Y
```

```
MPUT "oldprog.c" (y/n)? Y
```

```
MPUT "server.c" (y/n)? Y
```

```
200 PORT command okay.
```

```
150 Opening data connection for newcl.c (123.126.1.105,8557) (1333 bytes).
```

```
226 Transfer complete.
```

```
Transferred 1367 bytes in 0.03 seconds (50628.83 bytes/sec, 49.44 KB/s).
```

When FTP prompts you, you must respond Y(ES) or N(O) in uppercase or lowercase. FTP does not accept as a default.

When you send multiple files, the remote FTP server lists only the first file (newcl.c) before it displays the Transfer complete message.

When FTP prompts you, you must enter either a Y or an N in uppercase or lowercase. FTP does not accept as a default. If you enter the -NQ option with a wildcard, FTP transfers multiple files automatically.

► **NLST** [*remote-directory*] [*local-filename*]

Writes a list of the filenames and subdirectories in *remote-directory* in *local-filename*. If you omit *remote-directory*, the NLST command writes the list of files in your current working directory on your current directory. When you specify *local-filename* and omit *remote-directory*, insert a hyphen in the command line, as in the following example:

```
ftp> NLST - testfile
```

If you omit *local-filename*, PRIMOS FTP lists the filenames in *remote-directory* on the screen.

If *local-filename* already exists on the local host, FTP asks you whether you want to overwrite the existing file. If you answer yes, FTP creates a temporary file with a unique name and puts the list of files in it. Once the operation is completed successfully, FTP deletes the existing file and renames the temporary file with the filename that you originally requested. If you respond no, FTP suspends the transfer and displays the ftp> prompt. You can enter the command with a unique filename.

If the operation is interrupted, FTP displays a message informing you where the partially retrieved list of files is stored. If no data were transferred, the empty temporary file is deleted.

The NLST command is a synonym for the LS command.

Note

You can enter a wildcard name with the NLST command if the remote FTP server supports wildcard expansions. The wildcard symbols that you enter must conform to the wildcarding conventions that the remote host supports (for example, * for UNIX, or @ for PRIMOS systems).

► **OPEN** *sysname*

Establishes a connection to the FTP server on the system specified by *sysname*. *sysname* is either the remote system name or the remote system's Internet address (for example, 129.42.0.1). If you enter a name, that name can be in the HOSTS.TXT configuration file. If the name is not in the configuration file, the hostname server attempts to discover it. You can also enter an Internet address that is not in the HOSTS.TXT file if the host identified by the Internet address can be reached from the network to which your system is connected.

► **PUT** *local-filename* [*remote-filename*]

Stores *local-filename* on the remote system as *remote-filename*. When you omit *remote-filename*, PRIMOS FTP uses *local-filename* for the remote filename. The syntax of *remote-filename* must satisfy the requirements of the remote system.

If *remote-filename* already exists on the remote system, the file on the remote system may be overwritten, depending on how the remote system implements this command. To be certain that a file on the remote system is not overwritten, specify a unique *remote-filename*. Use the DIR, LIST, or NLST commands to list the current filenames.

PRIMOS FTP uses the current setting for file transfer representation type when it transfers the file. For example, if the file type is set to ASCII, PRIMOS FTP uses this setting.

The PUT command is a synonym for the SEND and STORE commands.

► **PWD**

Displays the pathname of the current working directory on the remote system.

► **QUIT**

Terminates a PRIMOS FTP session with the remote server, exits PRIMOS FTP, and returns to PRIMOS command level.

The QUIT command is a synonym for the BYE command.

► **RECV** *remote-filename* [*local-filename*]

Retrieves *remote-filename* from the remote system and stores it on the local 50 Series system as *local-filename*. If you omit *local-filename*, PRIMOS FTP uses *remote-filename* for the local filename. If you omit *local-filename*, *remote-filename* must be a filename, not a pathname. The syntax of *remote-filename* must satisfy the requirements of the remote system.

If *local-filename* already exists on the local host, FTP asks you whether you want to overwrite the existing file. If you answer yes, FTP creates a temporary file with a unique name and puts the contents of the retrieved file in it. Once the operation is completed successfully, FTP deletes the existing file and renames the temporary file with the filename that you originally requested. If you respond no, FTP suspends the transfer and displays the ftp> prompt. You can enter the command with a unique filename.

If the file transfer is interrupted, FTP displays a message informing you where the partially retrieved file is stored. If no data were transferred, the empty temporary file is deleted.

PRIMOS FTP uses the current setting for file transfer representation type when it transfers the file. For example, if the file type is set to ASCII, PRIMOS FTP uses this setting.

Note

When a binary file with an odd number of bytes is stored on a 50 Series system, PRIMOS adds a null character at the end of the file. When you send the file, FTP strips off the extra null character.

The RECV command is a synonym for the GET and RETRIEVE commands.

► **RENAME** *oldfilename newfilename*

Changes the names of a file on the remote system from *oldfilename* to *newfilename*. *oldfilename*, the name of the file that you want to change, may be specified as a pathname, but *newfilename* must be a filename.

► **RETRIEVE** *remote-filename [local-filename]*

Retrieves *remote-filename* from the remote system and stores it on the local 50 Series system as *local-filename*. If you omit *local-filename*, PRIMOS FTP uses *remote-filename* for the local filename. If you omit *local-filename*, *remote-filename* must be a filename, not a pathname. The syntax of *remote-filename* must satisfy the requirements of the remote system.

If *local-filename* already exists on the local host, FTP asks you whether you want to overwrite the existing file. If you answer yes, FTP creates a temporary file with a unique name and puts the contents of the retrieved file in it. Once the operation is completed successfully, FTP deletes the existing file and renames the temporary file with the filename that you originally requested. If you respond no, FTP suspends the transfer and displays the ftp> prompt. You can enter the command with a unique filename.

PRIMOS FTP uses the current setting for file transfer representation type when it transfers the file. For example, if the file type is set to ASCII, PRIMOS FTP uses this setting.

Note

When a binary file with an odd number of bytes is stored on a 50 Series system, PRIMOS adds a null character at the end of the file. When you send the file, FTP strips off the extra null character.

The RETRIEVE command is a synonym for the GET and RECV commands.

► **RMDIR** *remote-directory*

Deletes *remote-directory* on the remote system. You can only delete a remote directory that is empty (contains no files). To use the RMDIR command on an ACL directory of a remote 50 Series system, you must have Delete (D) access to that directory.

► **SEND** *local-filename* [*remote-filename*]

Stores *local-filename* on the remote system as *remote-filename*. When you omit *remote-filename*, PRIMOS FTP uses *local-filename* for the remote filename. The syntax of *remote-filename* must satisfy the requirements of the remote system.

If *remote-filename* already exists on the remote system, the file on the remote system may be overwritten depending on how the remote system implements this feature. To be certain that a file on the remote system is not overwritten, specify a unique *remote-filename*.

PRIMOS FTP uses the current setting for file transfer representation type when it transfers the file. For example, if the file type is set to ASCII, PRIMOS FTP uses this setting.

The SEND command is a synonym for the PUT and STORE commands.

► **STORE** *local-filename* [*remote-filename*]

Stores *local-filename* on the remote system as *remote-filename*. When you omit *remote-filename*, PRIMOS FTP uses *local-filename* for the remote filename. The syntax of *remote-filename* must satisfy the requirements of the remote system.

If *remote-filename* already exists on the remote system, the file on the remote system may be overwritten, depending on how the remote system implements this command. To be certain that a file on the remote system is not overwritten, specify a unique *remote-filename*.

PRIMOS FTP uses the current setting for file transfer representation type when it transfers the file. For example, if the file type is set to ASCII, PRIMOS FTP uses this setting.

The STORE command is a synonym for the PUT and SEND commands.

► **STRUCTURE** *structure-name*

Defines the way data is represented when it is transferred. *structure-name* is the name of the data structure. At Release 2.0, you must specify FILE structure. FILE structure is the default and is the only structure supported.

► **TYPE** *type-name*

Sets the file transfer representation type to *type-name*.

At Release 2.0, you can specify one of the following types with the TYPE command:

- ASCII — Sets the file transfer representation type to ASCII. Use ASCII to transfer text files. ASCII is the default type.
- Image — Sets the file transfer representation type to binary so that image files can be transferred.

Note

At Release 2.0, PRIMOS FTP does not support the transfer of the EBCDIC representation type.

If you do not specify a type, the current type is displayed. The default type is ASCII. ASCII and Image (or binary) are the only types that the current version of PRIMOS FTP supports. Specify type by a single character (A for ASCII and I for Image) as in the following example:

```
ftp> TYPE I
ftp> TYPE A
```

► USER *username*

Identifies you to the remote FTP server so that you can access its file system. *username* is a user ID that may be required by the remote system.

If the remote server also requires a password or account number, you are prompted after local echo and ECL are disabled. When FTP prompts you for *username*, it displays your local user ID as the default.

Accessing PRIMOS FTP From a Remote System

This chapter provides the following information for a user on another vendor's remote system who wants to transfer files to and from a 50 Series host:

- Access rights for a remote user and PRIMOS FTP server process
- FTP commands supported by the PRIMOS FTP server process on a 50 Series system
- How to enter a project ID
- How to transfer a file to a password-protected directory

Before You Transfer a File

Before you can send a file to a 50 Series system or retrieve a file from a 50 Series system, you must be certain that the PRIMOS FTP server process is running on the 50 Series system. The FTP server process listens for a request from a remote user and spawns a phantom to handle the request. Both you and the .TCP_FTP\$ ACL group, to which the PRIMOS FTP server belongs, must have certain file access rights to complete an FTP operation.

Note

At Release 1.0 of FTP (WSI300), the FTP server belonged to the .WSI_FTP\$ ACL group. To ease the transition for users of Release 1.0 (WSI300), the current FTP server is a member of both .WSI_FTP\$ and .TCP_FTP\$ ACL groups. A user of Release 1.0 can transfer files without changing access rights.

It is your responsibility to contact the System Administrator of the 50 Series system to set access rights.

Chapter 3, Using PRIMOS FTP on a 50 Series System, describes Prime access rights and how these rights are set on the 50 Series system. See the *PRIMOS User's Guide* for information on setting ACLs in general.

Access Rights for Remote System Users and the .TCP_FTP\$ ACL Group

If you want to send a file to a remote 50 Series system, both you and the .TCP_FTP\$ ACL group to which the PRIMOS FTP server process belongs need the following rights on the remote 50 Series system:

- Add (A) and Use (U) access to the directory that holds the file
- Read (R) and Write (W) access to files in the directory
- Use (U) access to any parent directories

Note

You and the .TCP_FTP\$ ACL group need Write (W) access so that the FTP server can overwrite a file if the file already exists.

To retrieve a file from a 50 Series system, both you and the .TCP_FTP\$ ACL group need the following rights:

- Read (R) access to the file
- Use (U) access to the target directory on the Prime remote host

If you want to delete a file on the remote 50 Series system, both you and the .TCP_FTP\$ ACL group must also have Delete (D) rights to the file. If you want to list the contents of a directory on the remote 50 Series system, both you and the .TCP_FTP\$ ACL group must have List (L) and Use (U) rights to the directory.

Table 5-1 summarizes access rights that a remote user and the PRIMOS FTP server process need to send, retrieve, and delete files, and to list the contents of a directory on a 50 Series system.

Table 5-1
Access Rights for Remote Users and PRIMOS FTP Server Process

<i>Access Rights</i>	<i>Operation</i>
ARUW	Send a file
RU	Retrieve a file
DU	Delete a file
LU	List directory

If the remote user is on a 50 Series system and is sending files to another 50 Series system, the user must have additional access rights on the 50 Series system from which files are being transferred. Table 5-2 summarizes those additional access rights.

Table 5-2
Access Rights for Invoking Server FTP from a Remote 50 Series

<i>Access Rights</i>	<i>Operation</i>
RU	Send a file
ADLUR	Retrieve a file
---	Delete a file
---	List directory (on screen)

Table 5-4, at the end of this chapter, lists the ACL rights that each PRIMOS server FTP command requires.

Project IDs

PRIMOS FTP supports project IDs for remote users connecting to 50 Series systems. On a 50 Series system, the System Administrator defines projects with the EDIT_PROFILE utility. See the *PRIMOS User's Guide* for a description of projects.

If you are required to enter a project ID, PRIMOS FTP displays the prompts when you connect to a 50 Series system, as in the following example:

```
ftp> open sys1

220 Prime Server FTP
username: smith
331 Enter PASS Command.
Password (sys1:smith):
332 Need project id for login, use ACCOUNT command
Project ID:

230 User logged in

ftp>
```

If you enter in response to the Project ID: prompt, you may be assigned to your own default project. If the System Administrator did not specify a default project and you enter , your attempt to log in fails.

If you enter a project ID or in response to the Project ID: prompt and the message *Insufficient access* appears, contact your System Administrator. The message indicates that the FTP server cannot access the System Administration Directory (SAD) on the 50 Series system. If project IDs were created after PRIMOS TCP/IP was installed, the System Administrator must reset the ACLs on the SAD.

Transferring Files From Password-protected Directories

PRIMOS FTP supports sending files to or retrieving files from password-protected directories on a 50 Series system. If you want to retrieve a file from or send a file to a password-protected directory on a 50 Series system, you must attach to the directory where the file either resides or is to be sent. When you want to transfer a file that resides in a subdirectory, you must first attach to the parent directory if the parent directory is protected by a password.

You can enter the CD command (or its equivalent) to attach to the password-protected directory, as in the following example:

```
ftp> CD TREE>BRANCH, SECRET
```

TREE>BRANCH is the pathname of a password-protected directory and SECRET is the password. After you attach to a password-protected directory, you can then enter the STOR or RETR command (or their equivalents).

When you use the CD command, as in the above example, a comma is required between the pathname (TREE>BRANCH) and the password (SECRET). A space before or after the comma is not permitted.

FTP Commands Supported by the PRIMOS FTP Server

This section describes the FTP commands supported by a PRIMOS FTP server process on a 50 Series system. The names of some FTP commands that your non-Prime system supports may differ from those described here. Consult your FTP manual for the syntax of FTP commands listed here. Your local system automatically translates and sends your commands to the PRIMOS FTP on the remote 50 Series system with the appropriate FTP server command name.

Note

You cannot enter any commands while a file transfer that you initiate is in progress. The FTP server process on the remote 50 Series host cannot accept any other commands until the current file transfer is completed.

Description of FTP Commands

The commands that the PRIMOS FTP server supports are listed below in alphabetical order. Table 5-3, at the end of this chapter, lists the equivalent PRIMOS FTP user commands on the 50 Series system.

► **ABORT**

Terminates an FTP session and exits FTP.

► **ACCOUNT *project-id***

Prompts you to enter *project-id* when a 50 Series system requires a project ID to log in. See the earlier section entitled Project IDs for an example of the prompts displayed.

► **CDUP**

Changes the parent directory of your current working directory.

► **CWD *remote-directory***

Changes the current working directory on the 50 Series system to *remote-directory*. The command is similar to the PRIMOS ATTACH command (described in the *PRIMOS Commands Reference Guide*). *remote-directory* must be a full pathname. The CWD command does not alter transfer parameters that you might have entered before you invoked the command.

► **DELE *remote-file***

Deletes *remote-file* from the working directory on the remote 50 Series system.

► **LIST *pathname***

Sends a list of all attributes of files in *pathname* from the remote 50 Series system to your local system. If you omit *pathname*, you are sent information about your current working directory on the 50 Series host. If *pathname* is a filename, the LIST command sends a list of the attributes of the file specified. The LIST command displays information similar to that displayed by the PRIMOS LD -DETAIL command.

The LIST command supports PRIMOS wildcarding.

► **MKD *pathname***

Creates *pathname* as a directory (if you specify a full pathname) or a subdirectory (if you specify a relative pathname) on the remote 50 Series system. A relative pathname must be preceded by **>*. The symbol ***, as the first directory in a pathname, designates everything in the pathname down to and including the current directory. For example, the command MKD **>BRANCH>LEAF* creates the subdirectory LEAF within the BRANCH directory. For more information on PRIMOS pathnames, see the *PRIMOS User's Guide*.

► **MODE *mode-name***

Sets the file transfer mode to *mode-name*. PRIMOS FTP server process supports the transfer of data as a stream of bytes (STREAM mode). STREAM mode is the default mode and is the only mode supported.

► **NLST *pathname***

Sends a list of all files in *pathname* from the remote 50 Series system to your local system. If you omit *pathname*, you are sent information about your current working directory on the 50 Series host. If *pathname* is a filename, the NLST command lists the file specified.

The NLST command supports PRIMOS wildcarding.

► **NOOP**

Causes the PRIMOS FTP server to send an OK, prompt. This command does not affect any parameters or previously entered commands.

► **PASS *password***

Specifies your password. *password* is a PRIMOS login password. The PASS command must be preceded by the USER command.

► **PORT *host-address port-address***

Specifies a data port in a data connection. *host-address* is an 8-bit Internet host address. *port-address* is a 16-bit TCP port address. In most instances, your system sends the PORT command automatically and you are not required to enter it.

► **PWD**

Displays the *pathname* of the current working directory on the remote 50 Series system.

► **QUIT**

Terminates an FTP session with the PRIMOS FTP server and exits FTP.

► **RMD *pathname***

Deletes either a directory (if you specify a full *pathname*) or a subdirectory of the current working directory (if you specify a relative *pathname*). A relative *pathname* must be preceded by *>. The symbol *, as the first directory in a *pathname*, designates everything in the *pathname*.

down to and including the current directory. For example, the command `RMD *>BRANCH>LEAF` deletes the subdirectory LEAF from the directory BRANCH.

To use RMD on an ACL directory of a remote 50 Series system, you must have Delete (D) access to that directory.

For more information on PRIMOS pathnames, see the *PRIMOS User's Guide*.

► **RNFR *old-filename***

The RNFR (rename from) command specifies the name of a file on the remote 50 Series system that is to be changed. *old-filename* is the name of the file that you want changed. *old-filename* may be specified as a pathname. The RNFR command must be followed by the RNTD command; otherwise, the RNFR command is ignored.

► **RNTD *new-filename***

The RNTD (rename to) command changes the name of a file specified in the RNFR command to *new-filename*. *new-filename* may be specified as a pathname. The RNTD command must be preceded by the RNFR command. Using the two commands together renames a file. Most FTP programs generate an RNFR command followed by an RNTD command in response to a single command that you enter.

Note

Most FTP implementations provide a rename command that generates both an RNFR and RNTD command.

► **RETR *remote-filename***

The RETR command retrieves a copy of *remote-filename* from the 50 Series system.

► **STOR *remote-filename***

Causes the 50 Series system to accept data from the user and store it as *remote-filename*.

► **STRU *structure-name***

Specifies the data structure indicated in *structure-name*. The PRIMOS FTP server process supports FILE structure. FILE structure is the default.

Note

At Release 2.0, PRIMOS FTP supports only FILE structure.

► **TYPE *type-name***

Sets the file transfer type to *type-name*.

At Release 2.0, the PRIMOS FTP server process supports the following types with the TYPE command:

- ASCII — Sets the file transfer type to ASCII. ASCII is the default type.
- Image — Sets the file transfer type to binary so that binary files can be transferred.

Enter either A or I with the TYPE command to set the file transfer type.

► **USER *username***

Identifies you to the PRIMOS FTP server process so that it can access the 50 Series system file system. *username* is a PRIMOS login user ID. The PASS command must follow the USER command.

Summary of User and PRIMOS FTP Server-supported Commands

Table 5-3 lists the commands that the PRIMOS FTP server supports for remote users and the equivalent PRIMOS FTP commands for 50 Series system users. Table 5-4 lists the ACL rights a remote user and the .TCP_FTP\$ ACL group need to invoke PRIMOS FTP commands.

Table 5-3
PRIMOS FTP Remote User Commands and Equivalent PRIMOS FTP
Server-supported Commands

<i>Remote 50 Series User Commands</i>	<i>PRIMOS FTP Server Commands</i>
	CDUP
BYE	ABORT
BYE	QUIT
CLOSE	QUIT
CONTROL-P	QUIT
QUIT	ABORT
QUIT	QUIT
CD	CWD
DELETE	DELE
GET	RETR
LIST	LIST
NLST	NLST
LOG	USER (if required) PASS (if required)
MKDIR	MKD
MODE	MODE
PORT	PORT
PWD	PWD
RMDIR	RMD
RENAME	RNFR RNTD
SEND	STOR
STRUCTURE	STRU
TYPE	TYPE
BINARY	TYPE I
IMAGE	TYPE I
ASCII	TYPE A
USER	USER (if required) PASS (if required)

Table 5-4
Access Rights for Server FTP Commands

<i>Server FTP Command</i>	<i>Remote User and .TCP_FTP\$ Access Rights</i>
CD	U
DELETE	DU
GET	RU
LIST	LU
MKDIR	ALU
NLST	LU
PWD	U
RENAME	ADLUR
RMDIR	DU
SEND	AU (if file does not exist) RUW (if file exists)

Note

You need Use (U) access to any parent directories to execute each of the above commands. If you want to execute the LIST command in a directory other than in the one to which you are remotely attached, you need List and Use (LU) access to the parent directories of that directory.

PRIMOS TCP/IP MAIL

This chapter explains how to use the PRIMOS TCP/IP MAIL facility. MAIL enables you to send messages to and receive messages from Prime and other vendors' systems connected to your local network or the Internet.

You can send mail to a user on

- A 50 Series system on a LAN300 that supports MAIL
- A PRIMENET node that supports MAIL
- Another vendor's system that is connected to an IEEE 802.3 LAN and that supports the Simple Mail Transfer Protocol (SMTP)

You do not have to know a route to the recipient's system. MAIL chooses the route (over PRIMENET or over SMTP and a LAN300) and then sends the message over the path that it chooses.

This chapter describes how to perform the following tasks:

- Invoking TCP/IP MAIL
- Listing, reading, and saving mail sent to you
- Sending mail
- Sending ASCII files, non-ASCII files, and directories

Chapter 7, A PRIMOS TCP/IP MAIL Commands Reference, describes all the MAIL commands and how you specify them. Appendix C, MAIL Messages, lists and describes the error messages that MAIL can display.

Before You Start

Before you can receive mail from another PRIMOS TCP/IP user or from a user on a host running a comparable SMTP based product, you must

- Have a MAIL ID and a MAIL account
- Initialize your mailbox
- Be certain that the MAIL servers are running

MAIL IDs

To receive mail, you need a MAIL ID. On a 50 Series system, your MAIL ID is your PRIMOS login account name. If you do not have a PRIMOS account, see your System Administrator.

If you want to send mail to a user on another host, you must know the recipient's MAIL ID. A MAIL ID for a recipient takes the form of a host name separated from a user ID by a @.

For example, the MAIL ID of user FERD on remote host SYS1 is

Ferd@sys1

Host names cannot contain a period. RD-A02 is a valid host name; RD.A02 is not a valid host name. A period (.) is included in a host name when it is a delimiter of a **domain name**. A domain name is the name of a system on the Internet. In the following example,

SMITH@RIXT-A02.ATT.COM

RIXT-A02.ATT.COM is the domain name of a user on the Internet. The suffix .COM specifies a top-level domain that is a commercial company in the United States. Other top-level domains include

<i>Domain</i>	<i>Description</i>
.ARPA	The DARPA Internet
.EDU	Educational institutions in the United States
.GOV	United States government agencies
.bitnet	The BITNET
.cs.net	The computer science network maintained by BBN
.NSF.net	The network of the United States National Science Foundation
.PDN	Companies and services reachable via public data networks
.UUCP	The UNIX-to-UNIX copy network

Some examples of electronic mail addresses are

<i>Example</i>	<i>Meaning</i>
SMITH	User on your system
Jones@ab-a01	User on another host at your site
WOOL@CD-C03.NPA.COM	User at a commercial company
wesson@bullet.UUCP	User on the UUCP network

When messages must pass through mail gateway systems to reach non-Internet sites, mail addresses become very complicated. If you do not know the MAIL ID of a recipient or his electronic mail address, contact the recipient directly or see your System Administrator.

Note

When you send mail to other networks, be careful to reproduce correctly the case of addresses, especially user IDs in the .UUCP domain. On some UNIX systems, for example, user ID wesson is not the same person as user ID WESSON.

Initializing Your Mailbox

You must enter the MAIL command once to initialize your mailbox after your System Administrator installs the MAIL software. You cannot be informed that you have received mail until you initialize your mailbox. You cannot send mail to other users who have not initialized their mailboxes.

Your mailbox is protected by your ACL rights. Only you and the MAILER_DAEMON, the process that picks up and delivers mail to you, can access your mailbox.

In the following example, user JONES enters the MAIL command to initialize her mailbox:

```
OK, MAIL
[MAIL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
(creating mailbox for Jones...)
No mail.
```

You can initialize your mailbox only if you are also allowed to create password directories or if your System Administrator has created a mailbox for you. The following messages indicate that your system does not permit the creation of password directories:

```
OK, MAIL
[MAIL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
(creating mailbox for Jones...)
Error code 327 from routine crepw$
Error code 10 from routine ac$set
```

If MAIL displays the above messages, contact your System Administrator. If you cannot initialize your mailbox, you can send mail, but you cannot receive mail.

MAIL Servers

Before you send mail, check to see (with the STATUS USERS command) whether the following MAIL servers are running:

- MAILER_DAEMON — The PRIMOS TCP/IP MAIL server that manages your request.

- SMTP_SENDER0 — The server that takes messages from the MAILER_DAEMON and passes them to the network.
- SMTP_SERVERnn — The server that receives mail messages from the network and passes them to MAILER_DAEMON. nn is the device address of the controller on which the server runs.

If any of these processes is not running, contact your System Administrator.

Using PRIMOS TCP/IP MAIL

You can invoke MAIL from PRIMOS command level (from the OK, prompt) or you can issue interactive commands when the * prompt appears within the PRIMOS TCP/IP MAIL program. Commands that act on a specific message must be invoked within MAIL. You must invoke MAIL to delete, file, or return a message that you have received.

The next sections illustrate how to invoke and use MAIL.

Listing Messages

You can list your messages from PRIMOS command level or you can invoke MAIL and act upon any messages that you have received.

For example, to list your mail messages, enter the following command at PRIMOS command level:

```
OK, MAIL -LIST
1   FERD           15 Sep 88 17:01  Meeting canceled
2   NICK_C         22 Sep 88 10:21  Report distributed
OK,
```

All MAIL messages are listed with a message number, the MAIL ID of the person who originated the message, the date and time when the message was sent, and the subject of the message.

To list your messages and then act upon them, the following alternative command issued at command level makes the * prompt appear:

```
OK, MAIL
1   FERD           15 Sep 88 17:01  Meeting canceled
2   NICK_C         22 Sep 88 10:21  Report distributed
*
```

The interactive commands that you can enter after the * prompt appears are described in Chapter 7, A PRIMOS TCP/IP Mail Commands Reference.

If you have no messages, MAIL responds:

```
No mail.  
OK,
```

Reading Your Messages

When you see the * prompt, you can select a message to read. Enter the number of the message and Return, or enter Return. Entering Return at the * prompt displays the first or next message in the list. In the following example, you enter 1 to read the first message:

```
OK, MAIL  
1  MFILMORE          10 Sep 88 10:01  Staff meeting  
2  MFILMORE          10 Sep 88 11:48  Meeting postponed  
* 1  
Return-path: <FERD@S3.Prime.COM>  
Received: (from user MFILMORE) by S10.Prime.COM; 22 Sep 88 11:49:39  
EDT  
Subject: Staff Meeting  
To:   FERD@S3.Prime.COM  
From: MFILMORE@S3.Prime.COM  
Date: 10 Sep 88 11:48 EDT
```

The staff meeting is scheduled for 10 AM.

*

The first two lines of this message are the message header, which shows the sender, date and time of the message, and other information. The message follows the header.

When you see the MAIL prompt (*), you can specify another message number or you can enter Return to see the next message in the list.

When you want to exit MAIL and return to PRIMOS command level, enter, in uppercase or lowercase,

Q

You see the PRIMOS OK, prompt.

Sending Mail

MAIL allows you to send mail at any time. If the network or the recipient's computer is not operational, PRIMOS TCP/IP stores your message and then forwards it when the network or computer comes back online.

This section describes how to send mail from either PRIMOS command level or from within PRIMOS TCP/IP and how to send a file from command level. You can send an ASCII (text) file, a non-ASCII file (binary file, runfile, or wordprocessing file), or a directory, segment directory or access category. If you send a non-ASCII file or a directory, you must encode the file or directory.

Sending Mail From PRIMOS Command Level: At PRIMOS command level, enter the MAIL command and the MAIL ID of the recipient. The following example shows a user sending mail to recipient SMITH on a remote host.

```
OK, MAIL SMITH@REMOTE-SYS
Subject: Your package
```

```
enter message, end with line containing only '$';
```

```
Your package has not arrived. Please inform me when it was mailed.
```

```
Thank you.
```

```
$
sending to SMITH ...
```

Note

MAIL does not support an editor. As you type your message, you can edit it only by erasing the last letters of a line and by retyping them. You cannot erase anything before a previous carriage return. The MAIL program supports sending files, which you can edit with EMACS or ED.

MAIL informs you that mail has been sent. If the recipient is on a 50 Series host, PRIMOS TCP/IP displays the following message to inform the recipient that new mail has arrived:

```
*** MAILER_DAEMON (user 105 on S12) at 16:34
new mail from RICH@REM-SYS
```

MAILER_DAEMON is the MAIL server running with a user ID number of 105 on the recipients 50 Series system. RICH@REM-SYS is the MAIL ID of the user who has sent the message.

Some tips to know as you type your message:

- Specify the address of a user on another system in the following format:

user-ID@hostname

where *hostname* is separated from the user ID by an @. If you do not know the recipient's user ID or the recipient's host, check a mail directory or check directly with the recipient.

- If you do not want a subject for your message, press Return at the Subject: prompt.
- You must type the \$, which terminates your input and sends your message, as the first character on a line of its own.
- Press Ctrl P if you want to prevent the message from being sent. MAIL places the message in a file named DEAD.LETTER in the directory of your initial attach point.

Sending an ASCII File From PRIMOS Command Level: The following command sends an ASCII file that you specify to one or more designated recipients:

MAIL [-FILE] *filename* [-TO] *address-1* [...*address-n*]

filename is a filename or a full pathname. *address-1* is the address of a recipient. When you use -FILE, you must use -TO or supply an address. -FILE and -TO can appear in any order on a command line. For example, the command

OK, MAIL -TO GEORGE@PIGEON BARBARA@RITZ -FILE JULY>REPORT

sends the file JULY>REPORT to recipients GEORGE and BARBARA.

You can omit either -FILE or -TO or both from the command line when you want to send a file to a recipient but, if they are omitted, *filename* must appear before *address*. The following examples show one or both options omitted from command lines that send the file REPORT to BARBARA@RITZ:

OK, MAIL REPORT BARBARA@RITZ

OK, MAIL -F REPORT BARBARA@RITZ

OK, MAIL REPORT -T BARBARA@RITZ

You also can issue the SEND command from within the PRIMOS TCP/IP MAIL program when the * prompt appears. For example, the MAIL -INTERACTIVE command, described in the next chapter, places you within PRIMOS TCP/IP MAIL.

Sending Directories and Non-ASCII Files from PRIMOS Command Level: MAIL enables you to send the following kinds of directories and non-ASCII files to a recipient on another 50 Series system running MAIL:

- Segment directories
- Access categories
- Runfiles
- Binary files
- Wordprocessing files

You must use the MAIL command to encode non-ASCII files and directories before you send them.

Use the following command to send a directory or non-ASCII file to one or more recipients:

MAIL -FILE *pathname* -TO *address-1* [*address-n*] -BINARY

pathname is a directory name, filename, or full pathname. *address-1* is the address of a recipient. You can omit either the -FILE or -TO or both options, but *pathname* must precede *address-1*. You must use the -TO option if you are mailing a directory to more than one address. The -BINARY argument is required. You can enter the options in any order.

You can send directories that contain non-ASCII files.

For example, the following commands mail a directory named SAMPLE.DIR to user ELMER:

```
OK, MAIL -FILE SAMPLE.DIR -TO ELMER@S1 -BINARY
```

```
OK, MAIL -BINARY SAMPLE.DIR ELMER@S1
```

```
OK, MAIL SAMPLE.DIR -TO ELMER@S1 -BINARY
```

Saving MAIL

Use the FILE command to copy messages from your mailbox and put them in a file. The following example shows how to save mail.

```
OK, MAIL
1  BILLY.M           08 Jun 88 10:23  Career change
2  EDG               13 Jun 88 10:28
* FILE STATUS.REPORT 1 2
```

In the example, STATUS.REPORT is the name of the file to which both messages are copied.

If you specify an existing filename or pathname, MAIL displays the following message:

```
STATUS.REPORT already exists, overwrite it?
```

The APPEND command allows you to append messages to an existing file. In the following example, you append both messages to a file named STATUS.REPORT:

```
OK, MAIL
1  BILLY.M          08 Jun 88 10:23  Career change
2  EDG              13 Jun 88 10:28
* APPEND STATUS.REPORT 1 2
```

If you specify a filename or pathname that does not exist, MAIL displays the following message:

```
STATUS.REPORT doesn't exist, create it?
```

Saving a Non-ASCII File or Restoring a Directory

To save a non-ASCII file or to restore a directory sent to you, perform the following steps:

1. Attach to the partition in which you want to restore the file or directory.
2. Invoke MAIL.
3. Display the MAIL message that contains the encoded file or directory.
4. Enter the FILE command without any options.

An Example of Restoring a Directory: The following example shows how to restore a mailed directory:

```
OK, MAIL
1  HARRY            23 Oct 88 22:12
2  DEWEY            24 Oct 88 10:10
3  MELVIN           24 Oct 88 11:14
* 3
Return-path: <MELVIN@S1.bates.COM>
Received: (from user MELVIN) by S2.Prime.COM; 24 Oct 88 10:14:50 EDT
Encoding: FS
To:   EARP@rem.sys.Prime.COM
From: MELVIN@S1.bates.COM
Date: 24 Oct 88 11:14:50 EDT

[ directory sample.dir
[ file PREFACE.BOOK
type SAM
[ data 1398 16 SUM
```

```
07669P,WAZ^7;U-#S$/F[_+MX?2LYN_R[:#,K-30UN7R\^GO[J"QKK"PW8HP,3E;NG
CY=O,Y[&RL+#=BHHP-/T\NGNY]O$S\/.]>WBY?*]HKS.]>WBY?*^HJR*D0C2]>[NZ>[
GR.7AY+VBO-3E^/2^HJR*D0C2]>[NZ>[GQN_O]+VBO-3E^/2^HMW]B#P^_M[>7N]/OT
_3RZ>[GV]3/P\+O[^O$Y?#TZ+VBO,/OY.6^HJR*D1'4S\/#Z.'P].7RQ.7P].B]HKS#
[^3EOJ*LBI$1QNGR\_3#[^ [TY>[T\]#AY^6]HKS.]>WBY?*^HJR*D1'&Z?+S]-#RY>;
AX^70X01734dY^6]HKS.]>WBY?*^HMW]BHH P,3I\^'B[.7;S_7T[.GNY:S4X>+LY<
SI\_2LQNGG]?+ES.GS]-V* ,##[^WMY>[T^\#$Z?/AXN--- more ---Q
```

*** FILE**

Decoding ...

start directory <TOP>MIDDLE>BOTTOM>sample.dir

end directory <TOP>MIDDLE>BOTTOM>sample.dir

*** Q**

Summary: Keep the following points in mind:

- Use the same procedure to restore both a non-ASCII file and a directory.
- Always enter the FILE command without any options when the * prompt appears.
- You cannot change the name of a non-ASCII file or directory mailed to you. Note that in the above example the name of the directory is contained in the encoded file.
- You cannot overwrite an existing non-ASCII file or directory. If you attempt to restore a non-ASCII file or directory that already exists, MAIL displays the following message:

*** FILE**

Decoding ...

object <TOP>MIDDLE>BOTTOM>sample.dir already exists

- You can restore an encoded non-ASCII file or directory only if it is stored in your mailbox. You cannot restore an encoded non-ASCII file or directory that has been saved with the APPEND or the FILE command.

Caution

When you restore a directory, do not enter a directory name with the FILE command at the * prompt. If the directory name that you enter already exists as a subdirectory of the current attach point, MAIL searches all visible partitions and attempts to restore the encoded directory in a top-level directory that matches the name that you entered. If the name that you enter with the FILE command does not exist, MAIL copies the directory into that file.

See Chapter 7, A PRIMOS TCP/IP MAIL Commands Reference, for a complete description of all MAIL commands that you can issue from PRIMOS command level or from within MAIL.

PRIMOS TCP/IP MAIL Help Facility

MAIL has an online help facility. Enter MAIL -HELP at PRIMOS command level or HELP (abbreviated H) from within MAIL to see a list of MAIL commands. In the following example, you issue the HELP command from PRIMOS command level.

```
OK, MAIL -HELP
[MAIL Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
[MAIL Version 2.0.0]
Usage:
    mail <filename> <user@host> <user@host> ...;
    mail <filename> !<mailing-list>
    mail <username>          -- accepts message from terminal
    mail                    -- list and read mail
    mail -List
    mail -Help
    mail -Into              -- enters interactive mode always
    mail -Delete_Mailbox
    mail -Set_Forward_to <address>
    mail -Cancel_Forwarding
    mail [-To <addresses>] [-File <filename>] [-Subject ...]
        [-Certify <to>] [-CC <addresses>]
        [-Binary] [-Alias <alias-file-name>]

    send problem reports to Postmaster@<system-name>
    specified file.
```

To display information about a command, enter the name or abbreviation of a specific command with the HELP command.

For example,

```
* HELP SEND
```

displays information about the SEND command.

If Mail Cannot Be Delivered

Mail cannot be delivered if the user ID is not known, the address is not known, or the remote host is down. The next sections describe some of the more common problems and errors.

Appendix C, MAIL Messages, lists and describes all MAIL error messages that can be returned to you.

Unknown User ID

If you send a message to a user on the local system and MAIL cannot match the user ID that you specified, it notifies you with a message similar to the following example:

```
Failed to send to: user-id
Service said: 550 Cannot write to mailbox; user not known
Message saved in DEAD.LETTER
```

If you send a message to an unknown user ID on your host, MAIL notifies you immediately. *DEAD.LETTER* is the pathname of a file in the directory of your initial attach point. The *DEAD.LETTER* file contains a copy of your returned message.

If you sent mail to an unknown user ID on another host, you are notified by the MAILER_DAEMON that MAIL attempted to locate the recipient and then returned the message to you. In that case, the message from the MAILER_DAEMON appears in your list of MAIL messages, as in the following example:

```
1 MAILER-DAEMON 04 October 88 09:45 returned mail; 550 Cannot write to
mailbox; user not known
```

Unknown MAIL Address

When you enter an incorrect MAIL address, MAIL returns the following message to you:

```
Cannot accept; domain not known
```

Check the way you typed the host name. A common mistake is omitting the hyphen (-). For example, the host name AB.a01 is incorrect.

Host Unavailable

If MAIL cannot send your message because of a problem involving the communication line or the destination host, it attempts to send your mail for a specified period (default of 3 days). Then it gives up and returns the message as mail from MAILER_DAEMON.

A PRIMOS TCP/IP MAIL Commands Reference

This chapter describes all PRIMOS TCP/IP MAIL commands and the way to specify them. The first part of the chapter describes commands that you issue at PRIMOS command level. These commands enable you to manage your mailbox and send mail. The second part of the chapter describes the interactive commands that you issue within MAIL (indicated by the asterisk (*) at the left margin).

MAIL Commands at PRIMOS Command Level

The commands described in this section are the only MAIL commands that you can enter at PRIMOS command level.

The MAIL command has the following format:

MAIL [*subcommand*] [*options*]

The MAIL command issued without any subcommands lists your messages and then gives the MAIL prompt *. Otherwise, you see the response `no mail` and then the PRIMOS OK, prompt. The MAIL command cannot be abbreviated.

Some MAIL subcommands enable you to manage your mailbox; others enable you to send mail. Subcommands for managing your mailbox are described in the next section.

MAIL Subcommands to Manage Your Mailbox

The following MAIL administrative subcommands are listed in alphabetical order. If the subcommand has an abbreviation, the abbreviation is placed immediately below the full form.

▷ **-CANCEL_FORWARDING**
-CF

Returns mail to your MAIL address, canceling the directive that you issued with the **-SET_FORWARD** subcommand described below.

▷ **-DELETE_MAILBOX**
-DM

Deletes your mailbox after it is empty. You must dispose of all messages before this command takes effect.

If you invoke MAIL after typing this command, your mailbox is recreated. Thus, you can restore your mailbox in case of an accidental deletion.

▷ **-HELP**
-H

Lists all the MAIL subcommands and options that you can issue at PRIMOS command level.

▷ **-INTERACTIVE**
-I

Places you within MAIL. It prompts you with * after listing your messages.

This subcommand works the same as the MAIL command, unless you have no messages. To be certain of entering MAIL interactively, use this option instead of MAIL.

▷ **-LIST**
-L

Lists your MAIL messages and then returns you to the OK, prompt at PRIMOS command level. By adding MAIL -LIST to your LOGIN.CPL file, you can see a list of your messages whenever you log in.

▷ **-SET_FORWARD_TO *address***
-SF

Forwards your mail to the address that you specify. *address* is any valid network address. You can use this command when you are moving to another system at your site or when you want another user to handle your mail temporarily.

Subcommands to Send Mail

The MAIL command (with subcommands) to send messages from PRIMOS command level has the following full format:


```
MAIL [ -FILE file_system_object
      -TO address-1 [address-n]
      -BINARY
      -SUBJECT text
      -CC address-1 [address-n]
      -CERTIFY [address]
      -ALIAS alias_filename ]
```

The following subcommands, which you can use in any order, are listed in alphabetical order:

▷ **-ALIAS *alias_filename***
-A

Sends your message to alias names of MAIL users. An alias is a shortened name that you enter into a command line without typing a complete MAIL ID. A MAIL address for remote site recipients can be quite long.

The aliases and their corresponding MAIL addresses are specified in a file, which you create. You may include any information (for example, phone number, mail stop) in a file entry as long as only one @ is present. Only the text immediately before and after an @ and delimited by white space is isolated and taken as the address. The text may be surrounded by angle brackets <>. An address surrounded by parentheses () is ignored even if that text contains an @.

The following example is an alias file:

```
Bill Rodney WmRD@Ab-a-01
Toni Wakefield 611; 665-4449 AnWK@Cd-c02.HIJ.COM
Dick Simpson Simp@.COM
```

The only information that MAIL reads as addresses from the above file is WmRD@Ab-a-01 and AnWK@Cd-c02.HIJ.COM.

Specify the alias filename on the MAIL command line. Use the -TO prefix before the aliases. Each entry after -TO is read as a separate user ID.

The following MAIL command sends the file REPORT to Bill and Toni, users listed in the alias file named MYALIASFILE:

```
OK, MAIL REPORT -TO BILL TONI -ALIAS MYALIASFILE
```

If you omit the -ALIAS option from the command line, MAIL looks for the file MAIL>MAIL_NAMES.ABV in the directory that is your initial attach point. You can create a file of this name using the same format as for any other alias file. Then, this file is read when you do not specify the -ALIAS option.

The following example does not specify the -ALIAS but relies on MAIL to check the default alias file:

```
OK, MAIL REPORT -TO BILL TONI
```

▷ **-BINARY**
-B

Enables you to mail directories (including segment directories and access categories) and non-ASCII files (including binary files, runfiles, and wordprocessing files) to a recipient on a 50 Series system. You can use the **-BINARY** option with the **-TO** and **-FILE** options. If you omit either **-TO** or **-FILE** or both, you must supply a directory name, filename, or pathname and an address. If both **-TO** and **-FILE** are omitted, the directory or filename must appear before the address. You must include **-TO** when mailing a non-ASCII file or directory to more than one address.

Note

You can mail a non-ASCII file or directory only to a recipient on a 50 Series system. Only a PRIMOS TCP/IP MAIL program can restore an encoded non-ASCII file or directory.

The following commands mail a binary file named TEST.BIN to user FRED:

```
OK, MAIL -FILE TEST.BIN -TO FRED@rem.sys -BINARY
```

```
OK, MAIL TOP>TEST.BIN FRED@rem.sys -BINARY
```

```
OK, MAIL TEST.BIN -TO FRED@rem.sys -B
```

```
OK, MAIL -B TEST.BIN FRED@rem.sys
```

Use the **FILE** command (described below) to restore a non-ASCII file or directory that you receive on a 50 Series system. You can save a non-ASCII file or restore a directory only when MAIL displays the interactive prompt (*).

▷ **-CC *address-1* [*address-n*]**

Sends copies of your message to specified addresses.

address-1 is the address of the first recipient; *address-n* are the addresses of other recipients.

▷ **-CERTIFY [*address*]**
-CE

Tags the recipient's message with a certification notice; the MAILER_DAEMON informs you or another user when the message was sent. Then, you or another user are notified when the message reaches a relay node and when the recipient reads, prints, files, or deletes the message.

address is the MAIL ID of the user who receives the certification notice. *address* can be the address of someone other than you.

Enter `-CERTIFY` before the address that is to receive the certification notice. If you include `-CERTIFY` on the command line without an address, MAIL notifies you when the recipient has received the notification.

In the following example, the file REPORT is sent to one user and the certification notice is sent to another:

```
OK, MAIL REPORT GEORGE@PIGEON.acme.COM -CE BARBARA@RITZ
```

If the mail passes through one or more relay nodes, the MAILER_DAEMON sends *address* reports of its journey. In the following example, user BARBARA receives the following messages in separate notifications:

```
your mail to george@pigeon.feathers.COM
was accepted for relay by S1.acme.COM
on 27 Oct 88 15:19:26 EST

your mail to george@pigeon.feathers.COM
was forwarded to S12.Prime.COM
on 27 Oct 88 15:26:55 EST

your mail to george@pigeon.feathers.COM
was delivered to george@pigeon.feathers.COM
on 27 Oct 88 15:28:34 EDT

your mail of 27 Oct 88 15:28
re:
was read by GEORGE
on 27 Oct 88 15:30:03 Wednesday
```

Only a 50 Series system running PRIMOS TCP/IP MAIL can provide a certification response.

▷ `-FILE file_sys_obj`
`-F`

Sends the file system object that you specify to designated recipients. *file_sys_obj* can be a filename or the name of a directory, segment directory, or access category. *file_sys_obj* can be a pathname. If *file_sys_obj* is a non-ASCII file or a directory, segment directory, or access category, you must use the `-BINARY` option so that the recipient can restore the file system object as a non-ASCII file or a directory.

When you use `-FILE`, you must use `-TO` (described below) or supply an address. `-FILE` and `-TO` can appear in any order on a command line. If you are sending a file system object to more than one address, you must use the `-TO` option. For example, the command

```
OK, MAIL -FILE JULY>REPORT -TO GEORGE@PIGEON BARBARA@RITZ
```

sends the file JULY>REPORT to recipients GEORGE and BARBARA.

The following command sends a directory named REPORT to GEORGE and BARBARA:

```
OK, MAIL -FILE JULY>REPORT -TO GEORGE@PIGEON BARBARA@RITZ -BINARY
```

You can omit either -FILE or -TO or both from the command line when you want to send a file to a recipient but if they are omitted, *file_sys_obj* must appear before the address. The following examples show one or both options omitted from command lines that send the file REPORT or the directory SAMPLE.DIR to BARBARA@RITZ:

```
OK, MAIL TOP>REPORT BARBARA@RITZ
```

```
OK, MAIL -F REPORT BARBARA@RITZ
```

```
OK, MAIL REPORT -T BARBARA@RITZ
```

```
OK, MAIL SAMPLE.DIR BARBARA@RITZ -B
```

Note

You can only send ASCII (text) files using MAIL. Mailing binary files, runfiles, or wordprocessing files is not supported unless the file is mailed as part of a directory or encoded with the -BINARY option. If you send a directory, you must encode the directory using the -BINARY option.

You can mail a non-ASCII file or directory only to a recipient on a 50 Series system. Only another PRIMOS TCP/IP MAIL program can restore an encoded non-ASCII file or directory.

▷ *[filename] [!mailing_list_file1] [!mailing_list-file-n]*

Sends *filename* to addresses in *mailing_list_file*. Both *filename* and *mailing_list_file* may be pathnames.

You can specify two or more lists or addresses or both on a command line. When you specify more than two mailing lists, each list must be preceded by a !. The following command sends the file memo_alert to addresses in two files and two users:

```
OK, MAIL MEMO_ALERT -FILE !LIST1 !LIST2 ellis@rem-sys.far.away gove@pigeon
```

A mailing list file has one address per line. There is no limit to the number of addresses that a list can contain.

The following file is an example of a mail list file:

```
SMITH
JONES@AB-A01
ANDY@CD-C03.ABD.COM
FERD
DIANE@AB-AC2.NYU.TEL
```

If an entry does not contain an @, MAIL takes the entire line as the address. If the entry has an @, MAIL isolates the text immediately before and after the @ and delimited by white space and takes that text as the address. The text may be surrounded by angle brackets. An address surrounded by parentheses is ignored even if that address contains an @. For example, MAIL reads the address of the entry

```
ELLIS <ELLIS@rem-sys.far.away>
```

as ELLIS@rem-sys.far.away.

However, MAIL reads the entry

```
ELLIS ELLIS@rem-sys.far.away
```

as ELLIS, the information in parentheses being discarded.

▷ **-SUBJECT *subject***
-S

Enables you to specify a subject for your message on the command line instead of answering the Subject prompt within MAIL.

▷ **-TO *address [address-n]***
-T

You must use the -TO prefix with more than one address. You can specify as many recipients, or addresses, as you wish. The following example sends mail to user Jones who has two addresses:

```
OK, MAIL -TO JONES@AB-A01 HBB@AB-A02
```

You need not enter the -TO option when you have only one address.

PRIMOS TCP/IP MAIL Interactive Commands

This section describes MAIL interactive commands. Use these commands when MAIL displays the interactive prompt (*).

If you enter MAIL when you have some mail, the MAIL displays the * prompt. The MAIL -I command, issued from PRIMOS command level, places you within the program whether or not you have mail.

▶ Return

Entering Return in response to the * prompt displays the first message in your MAIL list. Enter Return again to display the next message.

► **message-number**

Enter *message-number* to display a specific message. For example,

```
* 3
```

displays message number 3.

► **APPEND *filename* [*message_number(s)*] [-ALL]**

A

Copies messages from your mailbox and puts them at the end of an existing file. *filename* is the name of the file to which you want to append your message. *filename* can be a pathname. *message_number(s)* is the number or numbers of one or more messages that you want to append to *filename*. -ALL specifies that all messages in your mailbox are to be copied to *filename*.

If you enter the APPEND command with the name of a file and enter Return, MAIL appends the last message that you read to that file. If you specify a file for your messages that does not exist, MAIL asks about creating it. The following example shows how to append the last message read to the MY.MAIL file, which does not exist:

```
* A MY.FILE
my.mail doesn't exist, create it? YES
*
```

The following example shows how to append message number 3 to the file JULY>MY.MAIL:

```
* A JULY>MY.MAIL 3
*
```

► **COPY [*address-1*] [*address-n*] [!*mailing_list_file*]**

C

Copies the last message that you read and sends it to designated recipients. You can append a response to the message before you send it. On the command line you can specify either one or more addresses or a file containing addresses or both.

address-1 is the address of the first recipient. *address-n* are the addresses of other recipients. *mailing_list_file* is a filename or full pathname that contains the addresses of recipients.

Specify your user ID on the command line or in the mailing list file to keep a copy of your response. If you omit an address or a mailing list filename, the COPY command sends the annotated message to the sender.

In the following example, the last read message is annotated and mailed to addresses contained in a mailing list file named MAIL_LIST.FILE.

```
* C !MAIL_LIST.FILE
```

(enter annotation, end with line containing only '\$')

```
Please review this report.
```

```
$
```

```
sending to BARBARA@RITZ ...
```

```
sending to WIGG@UNC
```

► DELETE [*message_number(s)*] [-ALL]

D

Deletes one, several, or all of your messages. MAIL does not automatically delete your messages when you print or file them. Messages are kept, by default, until you explicitly delete them.

message_number(s) is the number or numbers of one or more messages that you want to delete. -ALL deletes all messages in your mailbox.

In the following example, two specific messages (numbers 2 and 4) are deleted:

```
* D 2 4
```

The DELETE command followed by deletes the last message that you read.

You can recover messages that you accidentally delete because PRIMOS TCP/IP MAIL does not actually delete messages until you exit mail. See the UNDELETE command.

► FILE [*filename*] [*message_number(s)*] [-ALL]

F

Copies messages from your mailbox and places them in a file or restores a non-ASCII file or directory. Directories include segment directories and access categories. *filename* is the filename or full pathname to which messages are copied. *message_number(s)* is the number or numbers of one or more messages to be copied. If you omit *message_number(s)*, MAIL copies the last message that you read to *filename*. The -ALL option files all messages to *filename*.

If you specify an existing filename, MAIL asks about overwriting it.

In the following example, two messages (numbers 4 and 6) are copied to the file NOTE.TWO:

```
* F NOTE.TWO 4 6
```

To restore a non-ASCII file or directory sent to you, perform the following steps:

1. Attach to the partition in which you want to restore the file or directory.

2. Invoke MAIL.
3. Display the MAIL message that contains the encoded file or directory.
4. Enter the FILE command without any options.

The following example shows how to restore a mailed directory:

```
OK, MAIL
1  GENZ                25 Oct 88 22:12
2  GREEN              26 Oct 88 10:10
3  WOLPE              27 Oct 88 11:14
* 3
Return-path: <WOLPE@S1.Rabbit.COM>
Received: (from user WOLPE) by S2.Prime.COM; 24 Oct 88 10:14:50 EDT
Encoding: FS
To:   EARP@rem.sys.Prime.COM
From: WOLPE@S1.miflin.COM
Date: 27 Oct 88 11:14:50 EDT

directory sample.dir
[ file PREFACE.BOOK
type SAM
[ data 1398 16 SUM

07669P,WAZ^7;U-#$$\ /F[_+MX?2LYN_R[:#,K-30UN7R\^GO[J"QKK"PW8HP,3E;NGC
Y=O,Y[&RL+#=BHHP-/T\NGNY]O$$\/.]>WBY?*]HKS.]>WBY?*^HJR*D0C2]>[NZ>[GR
.7AY+VBO-3E^/2^HJR*D0C2]>[NZ>[GQN_O]+VBO-3E^/2^HMW]B#P^_M[>7N]/OT_3R
Z>[GV]3/P\+O[^O$Y?#TZ+VBO,/OY.6^HJR*D1'4S\/#Z.'P].7RQ.7P].B]HKS#[^3E
OJ*LBI$1QNGR\_3#[^ [TY>[T\]#AY^6]HKS.]>WBY?*^HJR*D1'&Z?+S]-#RY>;AX^7Q
X01734dY^6]HKS.]>WBY?*^HMW]BHH P,3I\^'B[.7;S_7T[.GNY:S4X>+LY<SI\_2LQ
NGG]?+ES.GS]-V* ,##[^WMY>[T^\#$Z?/AXN
--- more ---Q
* FILE
Decoding ...
start directory <TOP>MIDDLE>BOTTOM>sample.dir
end directory <TOP>MIDDLE>BOTTOM>sample.dir
* Q
```

Keep the following points in mind when you restore a non-ASCII file or directory:

- Use the same procedure to restore a non-ASCII file or to restore a directory.
- Always enter the FILE command without any options when the * prompt appears.
- You cannot change the name of a non-ASCII file or directory mailed to you. Note that in the above example the name of the directory is embedded in the encoded file.
- You cannot overwrite an existing non-ASCII file or directory. If you attempt to restore a file or directory that already exists, MAIL displays the following message:

*** FILE**

Decoding ...

object <TOP>MIDDLE>BOTTOM>sample.dir already exists

You can restore an encoded non-ASCII file or directory only if it is stored in your mailbox. You cannot restore an encoded file or directory if it has been saved with the APPEND or the FILE command.

Caution

When you restore a directory, do not enter a directory name with the FILE command at the * prompt. If the directory name that you enter already exists as a subdirectory of the current attach point, MAIL searches all visible partitions and attempts to restore the encoded directory in a top-level directory that matches the name that you entered. If the name that you enter with the FILE command does not exist, MAIL copies the directory into that file.

► HELP**H**

Gives you information about MAIL. Enter HELP to see a list of MAIL commands. Enter the name or abbreviation of a specific command with the HELP command to display information about that command. For example,

*** HELP D**

displays information about the DELETE command.

► LIST**L**

Lists the messages in your mailbox. The display reflects any messages added or deleted since the previous listing.

► PRINT [message_number(s)] [-ALL]**P**

Prints messages on your default device or a printer that you designate. *message_number(s)* is the number or numbers of one or more messages that you want to print. The *-ALL* option prints all messages in your mailbox. If you enter the PRINT command without any options and enter Return, PRIMOS TCP/IP MAIL prints the last message that you read.

You can designate a printer other than the default device on which to print the message. You also can use the PRIMOS Spool options, as long as you specify them at the end of the command line after the message number or the `-ALL` option. In the following example; message number 4 is spooled to printer QMSD:

```
* P 4 -AT QMSD
```

For a list of PRIMOS Spool options, see the *PRIMOS Commands Reference Guide*.

► **QUIT**
Q

Terminates your session with MAIL and returns the PRIMOS OK, prompt.

► **REPLY**
R

Enables you to reply to the last message that you have read. MAIL sends your reply to the originator. In the following example, the user reads message 1 and responds to that message:

```
OK, MAIL
1  FRAM          12 Sep 88 10:42  Meeting rescheduled
2  BARBARA       12 Sep 88 16:23  Book lost
* 1
Return-path: <@S2.Prime.COM:FRAM@REM-SYS.Prime.COM>
Received: from S51.Prime.COM by S2.Prime.COM; 12 Sep 88 11:02:31 EDT
Received: from REM-SYS.Prime.COM by S51.Prime.COM; 12 Sep 88 10:42:21 EDT
Received: from user FRAM; by REM-SYS.Prime.COM; 12 Sep 88 10:42:48 EST
Subject: Meeting rescheduled
To: bgilman@S2.Prime.COM
From: FRAM@REM-SYS.Prime.COM
Date: 12 Sep 88 10:42:48 EST
```

The meeting has been rescheduled for 2 PM tomorrow.

```
* R
```

enter message, end with line containing only '\$';

Thanks. I'll try to be there.

```
$
```

You can only respond to the last message that you read. If you specify a message number with the command or if you enter the command after the * prompt without having read a message, MAIL displays the following message:

No current message.

► **SEND** [*options*]
S

Sends mail to recipients. The SEND command has the same options as those for sending mail at PRIMOS command level. You can specify any of the options on the command line. Enter a space between each option and your specification.

The following summary describes each option. For a more complete description, see above.

<i>Option</i>	<i>Description</i>
-ALIAS <i>alias_filename</i>	Enables you to specify <i>alias_filename</i> , a file containing user IDs for aliases that you specify on the command line. You can type the name of your alias file as well as the aliases of the individuals.
-BINARY	Enables you to mail a non-ASCII file or directory, segment directory, or access category to a recipient on a 50 Series system. You must include the -TO and -FILE options or the filename or directory name and address of the recipient. You can only send a non-ASCII file or directory to a recipient on a 50 Series system.
-CC <i>address address-n</i>	Sends copies of your message to designated recipients.
-CERTIFY [<i>address</i>]	Notifies you or <i>address</i> when the message was sent and when the recipient read, filed, printed, or deleted it. <i>address</i> is the address of the person to be notified.
-FILE <i>file_sys_obj</i>	Sends <i>file_sys_obj</i> to specified addresses. <i>file_sys_obj</i> can be a filename or the name of a directory, segment directory, or access category. You can omit -FILE but then you must specify a filename or directory name before an address. If <i>file_sys_obj</i> is a non-ASCII file or directory, you must include the -BINARY option.
-SUBJECT <i>subject</i>	Enables you to specify a subject for your message on the command line instead of answering the Subject prompt within MAIL.
-TO <i>address address-n</i>	Sends a message to specified addresses. You can omit -TO unless you specify more than one address.

The following is an example of SEND command options:

```
* S -FILE THIS.FILE -CE -TO Smith@ab-a01 -CC Jones@ab-a01
```

In the above example, user ID Smith on host ab-a01 has been sent THIS.FILE along with a certify notice. User ID Jones, also on host ab-a01, has been sent a copy of the file. The originator is be notified when Smith receives and handles the message.

The SEND command without options displays the following prompts in the following order:

- FILE:
- TO:
- SUBJECT:

When FILE: is displayed, designate a file to send or press to create a file and display the next prompt. When TO: is displayed, enter an address or press to abort the SEND command and return you to the MAIL * prompt. At the SUBJECT: prompt, enter a description of your message. Enter and type \$ on a separate line to end the message.

► UNDELETE [*message_number(s)*] [-ALL]
U

Retrieves messages deleted within the current MAIL session. *message_number(s)* is the number or numbers of one or more messages to be restored. If you omit *message_number(s)*, MAIL restores the last message that you deleted. The -ALL option restores all messages deleted within the current MAIL session. The UNDELETE command followed by retrieves the last message that you deleted during the current session.

In the following example, the UNDELETE command restores message number 2:

```
* U 2  
*
```

Note

If you delete a message, you can read or spool that message to a printer by number without using the UNDELETE command. However, PRIMOS TCP/IP MAIL deletes this message permanently when you leave the MAIL program.

Part II: Programmer

Introduction to the PRIMOS Socket Library

This chapter introduces the PRIMOS socket subroutine calls. The PRIMOS socket subroutine library implements a subset of Berkeley UNIX 4.3BSD (Berkeley Software Distribution) subroutine calls. In addition to a standard set of socket subroutine calls, the PRIMOS socket library provides new subroutine calls to initialize the socket environment.

The PRIMOS socket implementation enables a user program on a 50 Series system to communicate with another application running on a remote host over an IEEE 802.3 network using the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). A 50 Series application must call the socket subroutines to access TCP and UDP protocols, which run on one or more LHC300 controllers.

The chapter includes a sample program that illustrates how to write a network application that calls the PRIMOS socket library.

The following topics are described in this chapter:

- Prerequisites for writing, compiling, and running a user program that calls the PRIMOS socket library
- Socket concepts
- Summary of the supported socket calls
- Usage
- Differences between PRIMOS and UNIX socket implementations
- Programming considerations
- Restrictions and limitations

Prerequisites

To run a program that incorporates PRIMOS socket library routines requires the installation of the following hardware and software on a 50 Series system:

- LHC300
- PRIMOS TCP/IP Release 2.0
- PRIMOS Rev. 22.0 or greater

- The PRIMOS C runtime libraries provided with Translator Family Release T1.0-21.0 or greater

To compile a program that incorporates PRIMOS socket library routines requires the PRIMOS C compiler and libraries.

In addition, you must configure PRIMOS TCP/IP and start the TCP/IP protocols on the LHC300. (For information about configuring PRIMOS TCP/IP, see Chapter 12; for information about starting the protocols, see Chapter 14.)

Socket Concepts

This section describes briefly the basic concepts of sockets that the PRIMOS socket implementation supports.

A **socket** is a data abstraction defined as an endpoint for communication. An application creates a socket and then binds a name to the socket. The application can then send and receive messages on the socket. In the socket model of communications, bound (named) sockets enable unrelated processes to communicate.

The most common form of socket communications is called the **client-server model**. In this model, the **server** process listens to a socket at one end of the data path. It accepts connection requests from client processes, receives messages, and sends replies. The **client** process initiates a connection with a listening server process. It communicates with the server process over another socket, created at the client's end of the data path.

Domains

Socket communications take place within a **domain** (also called a **family** or an **address format**). Two of the more important domains are the UNIX domain (**AF_UNIX**, address format UNIX) and the DARPA Internet domain (**AF_INET**, address format Internet). The PRIMOS socket library supports communication only within the Internet domain. (**AF_INET** is a symbolic constant defined in the `socket.h` include file.)

Socket Layers

The socket model consists of three layers (also called levels): the top socket layer, the middle protocol layer, and the bottom device layer. The socket layer is the interface to system calls. The protocol layer contains the protocols, such as TCP or UDP, that enable communication across the network. The device layer contains the programs that control network devices.

Socket Types

Sockets are classified by the abstract types that they support. Sockets of the same type can exchange information within a domain. The PRIMOS socket library supports only two types:

- Stream (SOCK_STREAM)
- Datagram (SOCK_DGRAM)

(SOCK_STREAM and SOCK_DGRAM are symbolic constants defined in the socket.h include file.)

The TCP protocol implements the SOCK_STREAM type. The UDP protocol implements the SOCK_DGRAM type. The PRIMOS socket library supports only TCP and UDP protocols.

SOCK_STREAM: SOCK_STREAM sockets are full-duplex byte streams. A SOCK_STREAM socket must be connected before a user program can send or receive data on the socket.

SOCK_STREAM protocols ensure that data transfers are not lost or duplicated. If data cannot be transmitted within a reasonable time, the connection is considered broken. Subsequent calls return an error condition.

TCP is an example of a SOCK_STREAM protocol that runs over IP and performs the reliability and sequencing for the higher layers.

SOCK_DGRAM: SOCK_DGRAM is a connectionless and unreliable message stream that allows full-duplex operation without the use of a connection. A higher layer protocol is expected to guarantee reliability and sequence the data flow.

UDP is an example of a SOCK_DGRAM protocol that claims no reliability.

Socket Names

Sockets that application programs create use socket names to refer to one another. Names are translated into addresses. Addresses are drawn from domains. An Internet domain name contains an Internet address and a port number.

A socket name is described internally by the sockaddr structure.

```
struct sockaddr {           /* Declared in socket.h */
    short  sa_family;       /* address format identifier */
    char   sa_data[14];    /* address */
};
```

The `sockaddr` structure is very general in order to preserve as much compatibility as possible across domains. It contains the following elements:

<i>Element</i>	<i>Description</i>
sa_family	Indicates to which domain the address belongs. All network addresses belong to one or more address families (domains). The domain defines the format and interpretation of the address. The PRIMOS socket library includes support only for the AF_INET (Internet) domain. The size of this field is 2 bytes.
sa_data	Contains the actual data value. The size of this field is 14 bytes.

In the AF_INET domain, the socket name is used to define the Internet address of the network interface to use and the port number on which the process expects data. User programs in the AF_INET domain use the more specific `sockaddr_in` structure, whose elements include the port number and Internet address:

```
struct sockaddr_in {      /* Declared in in.h */
    short                sin_family;
    unsigned short       sin_port;
    struct in_addr        sin_addr;
    char                 sin_zero[8];
};

struct in_addr {          /* Declared in in.h */
    unsigned long        s_addr;
};
```

The `sockaddr_in` structure contains the following elements:

<i>Element</i>	<i>Description</i>
sin_family	The same as <i>sa_family</i> in the <code>sockaddr</code> structure (2 bytes).
sin_port	The port number, which defines a particular socket user on the system (2 bytes).
sin_addr	The Internet address (4 bytes), an unsigned long integer that is defined as a structure for historical reasons. The Internet address defines the system to which the socket user is attached.
sin_zero	Eight empty bytes to make up the full 16 bytes in the <code>sockaddr</code> structure.

It is good programming practice to cast the `sockaddr_in` structure to the `sockaddr` structure in calls to socket subroutines. However, since the two structures are the same size, the cast is not required.

Association

The address and port number uniquely identify an Internet host process for a specific protocol. Communicating processes are bound by an association of domain names. An association consists of the following five items:

- Protocol
- Local address
- Remote address
- Local port
- Remote port

Associations are always unique and are never duplicated.

Half of the elements that comprise an association (local address, local port) are specified when a socket is bound. When a connection is established or messages accepted, a socket's association is completed.

Supported Socket Calls

This section summarizes all the socket calls that the PRIMOS socket library supports. Chapter 9, *Socket Subroutine Calls*, describes each subroutine call in detail.

The PRIMOS socket calls all begin with the prefix `TUP$`. If you want to use the customary UNIX names for the socket calls, include the file `redefine.h` in your program. `redefine.h` replaces the UNIX routine name with the corresponding PRIMOS name. For example,

```
#define accept TUP$accept
```

allows your program to call the routine `accept()` rather than `TUP$accept()`. The only exception is the socket routine `TUP$perror()`, which must be called by its full name so as not to be confused with the C library function `perror()`.

Summary of the Socket Environment Subroutines

A user program on a 50 Series system must incorporate subroutine calls specific to PRIMOS in order to initialize and manipulate the socket environment.

Table 8-1 lists and describes in alphabetical order the PRIMOS socket environment subroutines.

Table 8-1
Socket Environment Subroutines

<i>Subroutine</i>	<i>Description</i>
TUP\$close_socket_env()	Closes the socket environment.
TUP\$pass_socket()	Passes a socket to another process.
TUP\$receive_passed_socket()	Accepts a socket from another process.
TUP\$reset_socket_env()	Resets the socket environment.
TUP\$setup_socket_env()	Initializes the socket environment.

See the next section, entitled Usage, for a description of how to use these subroutines.

Summary of the Socket Subroutines

PRIMOS TCP/IP supports a subset of the Berkeley socket subroutines.

Table 8-2 lists and describes in alphabetical order the PRIMOS socket subroutine calls.

Table 8-2
PRIMOS Socket Subroutine Calls

<i>Subroutine</i>	<i>Description</i>
TUP\$accept()	Accepts an incoming connection request for a server's socket.
TUP\$bind()	Associates a name with a socket.
TUP\$close_socket()	Removes a socket entry.
TUP\$connect()	Initiates a client's connection to a server's socket.
TUP\$getpeername()	Returns the name of a socket's peer (the socket at the other end of the connection).
TUP\$getsockname()	Returns the current name for a socket.
TUP\$getsockopt()	Retrieves the values assigned to socket options.
TUP\$ioctl()	Performs control operations.
TUP\$listen()	Listens on a bound socket for connection requests from client processes.
TUP\$perror()	Displays an error message at the terminal.
TUP\$recv()	Receives messages from a connected socket.
TUP\$recvfrom()	Receives messages from an unconnected socket.
TUP\$select()	Monitors multiple I/O requests among multiple sockets.

Table 8-2
PRIMOS Socket Subroutine Calls – Continued

<i>Subroutine</i>	<i>Description</i>
TUP\$send()	Sends a message to a connected socket.
TUP\$sendto()	Sends a message to an unconnected socket.
TUP\$setsockopt()	Sets socket options.
TUP\$shutdown()	Shuts down all or part of a connection on a socket.
TUP\$socket()	Creates a socket.

See the next section, entitled Usage, for a description of how to use these subroutines.

Summary of the Network Library Subroutines

Network library routines provide auxiliary functions such as

- Mapping a host name to an Internet address
- Mapping Internet addresses to network numbers
- Obtaining controller device addresses

Table 8-3 lists and describes in alphabetical order the addressing and character manipulation subroutines that the PRIMOS socket library supports.

Table 8-3
Network Library Subroutines

<i>Subroutine</i>	<i>Description</i>
TUP\$getalladdrs()	Returns all the Internet addresses of a host when given any of the alternate names of the host.
TUP\$getbroadcastaddr()	Returns the broadcast address for a controller.
TUP\$getdevaddrs()	Returns the device addresses of LHC300 controllers.
TUP\$gethostbyaddr()	Returns information about the host when given an Internet address of the host.
TUP\$gethostbyname()	Returns information about the host when given an official name or alias name of the host.
TUP\$gethostname()	Returns the name of the local host.
TUP\$inet_addr()	Returns an unsigned long integer when given an Internet address in dot notation.

Table 8-3
Network Library Subroutines – Continued

<i>Subroutine</i>	<i>Description</i>
TUP\$inet_lnaof()	Returns a local address when given an Internet address specified as a structure.
TUP\$inet_makeaddr()	Returns an Internet address when given the network number and the local address specified as long integers.
TUP\$inet_netof()	Returns the network number when given an Internet address specified as a structure.
TUP\$inet_network()	Returns an unsigned long integer when given a network number in dot notation.
TUP\$inet_ntoa()	Returns an Internet address as a character string when given an Internet address specified as a structure.

See the next section, entitled Usage, for a description of how to use these subroutines.

Usage

This section describes how a network application incorporates the PRIMOS socket library routines.

The section contains two sample programs. The meaning of the socket calls is explained with reference to these programs.

An Example of a Stream Socket Program

The following sample programs consist of a client program on one 50 Series system and a server program on another 50 Series system. Each 50 Series system has one LHC300. The client creates a SOCK_STREAM socket (TCP protocol) to communicate with the server. The server listens indefinitely on a port (1101) for incoming requests from client programs. The client requests that the server return a character string. When the client receives the requested information, it closes the socket and the socket environment. The server closes the socket after it sends information to the client but does not close the socket environment.

The sample client and server programs issue socket subroutine calls in the following order:

<i>Client</i>	<i>Server</i>
TUP\$inet_addr()	TUP\$inet_addr()
TUP\$getdevaddrs()	TUP\$getdevaddrs()
TUP\$setup_socket_env()	TUP\$setup_socket_env()

TUP\$socket()	TUP\$socket()
TUP\$bind()	TUP\$bind()
	TUP\$listen()
TUP\$connect()	TUP\$accept()
TUP\$send()	TUP\$recv()
TUP\$recv()	TUP\$send()
TUP\$close_socket()	TUP\$close_socket()
TUP\$close_socket_env()	

The client program:

```
#include <stdio.h>
#include <in.h>
#include <inet.h>
#include <ioctl.h>
#include <netdb.h>
#include <socket.h>
#include <tcp/ip_error.h>
#include <tcp/ip_time.h>

main()
{
    unsigned long  address, TUP$inet_addr();
    struct dev_addr *dev_ptr, *TUP$getdevaddrs();
    CONTROLLER_LIST_ENTRY  controller_list[2];
    long  TUP$setup_socket_env(), TUP$socket(), TUP$bind(),
          TUP$connect(), TUP$send(), TUP$recv(),
          TUP$close_socket(), TUP$close_socket_env();
    long  cc;
    long  s;
    struct sockaddr_in  sock_name;
    struct sockaddr_in  rem_addr;
    static char  req[] = "Send me a message.";
    char  buff[80];

    address = TUP$inet_addr("193.44.5.4");

    dev_ptr = TUP$getdevaddrs(address);

    controller_list[0].controller_number = dev_ptr->dev_addr_array[0].dev_addr;

    cc = TUP$setup_socket_env(0, 1, controller_list, NULL);
```

```
s = TUP$socket(AF_INET, SOCK_STREAM, 0);

sock_name.sin_family = AF_INET;
sock_name.sin_port = 0;
sock_name.sin_addr.s_addr = dev_ptr->dev_addr_array[0].internet_address;

cc = TUP$bind(s, (struct sockaddr *)&sock_name, sizeof(sock_name));

rem_addr.sin_family = AF_INET;
rem_addr.sin_port = 1101;
rem_addr.sin_addr.s_addr = address;

cc = TUP$connect(s, (struct sockaddr *)&rem_addr, sizeof(rem_addr));

cc = TUP$send(s, req, strlen(req), 0);

cc = TUP$recv(s, buff, sizeof(buff), 0);

printf("Returned buffer - \n%s\n", buff);

cc = TUP$close_socket(s);

cc = TUP$close_socket_env();

}
```

The server program:

```
#include <stdio.h>
#include <in.h>
#include <inet.h>
#include <ioctl.h>
#include <netdb.h>
#include <socket.h>
#include <tcp/ip_error.h>
#include <tcp/ip_time.h>

main()
{
    unsigned long  address, TUP$inet_addr();
    struct dev_addrs *dev_ptr, *TUP$getdevaddrs();
    CONTROLLER_LIST_ENTRY controller_list[2];
    long  TUP$setup_socket_env(), TUP$socket(), TUP$bind(),
          TUP$listen(), TUP$accept(), TUP$recv(),
          TUP$send(), TUP$close_socket();
    long  cc;
    long  s;
    struct sockaddr_in  sock_name;
```



```
struct sockaddr_in rem_addr;
long rem_addr_len;
long ns;
char recvbuff[80];
static char ansbuff[] = "Sending message.";
static char qbuff[] = "No bytes in message buffer.";
char *cp;
long bufsize;

address = TUP$inet_addr("193.44.5.4");

dev_ptr = TUP$getdevaddrs(address);

controller_list[0].controller_number = dev_ptr->dev_addr_array[0].dev_addr;

cc = TUP$setup_socket_env(0, 1, controller_list, NULL);

s = TUP$socket(AF_INET, SOCK_STREAM, 0);

sock_name.sin_family = AF_INET;
sock_name.sin_port = 1101;
sock_name.sin_addr.s_addr = dev_ptr->dev_addr_array[0].internet_address;

cc = TUP$bind(s, (struct sockaddr *)&sock_name, sizeof(sock_name));

cc = TUP$listen(s, 5);

for(;;)
{
    rem_addr_len = sizeof(rem_addr);

    ns = TUP$accept(s, (struct sockaddr *)&rem_addr, &rem_addr_len);

    cc = TUP$recv(ns, recvbuff, sizeof(recvbuff), 0);

    if (cc > 0)
    {
        printf("Received buffer - %s\n", recvbuff);
        cp = ansbuff;
        bufsize = strlen(cp);

        cc = TUP$send(ns, cp, bufsize, 0);
    }
}
```

```
        else if (cc == 0)
        {
            cp = qbuff;
            bufsize = strlen(cp);

            cc = TUP$send(ns, cp, bufsize, 0);
        }
        else /* cc < 0 */
        {
            fprintf(stderr, "Error in TUP$recv() call.\n");
        }

        cc = TUP$close_socket(ns);
    }
}
```

The next sections describe how to use the PRIMOS socket calls.

Setting Up the Socket Environment in PRIMOS

To use either TCP or UDP on a 50 Series system that has one or two LHC300 controllers, call the `TUP$setup_socket_env()` routine and give it a list of the LHC300 controllers on which the protocols run. Before the user program can call `TUP$setup_socket_env()`, it should call `TUP$getdevaddrs()` to obtain the device addresses for the controllers on its local system.

Calling `TUP$getdevaddrs()`: The sample client and server programs both call `TUP$getdevaddrs()`, which has the following format:

```
cc = TUP$getdevaddrs(addr);
```

where *addr* is the Internet address of a host, in integer form. The client program gives `TUP$getdevaddrs()` the Internet address of the remote host to which it wants to connect. The server program gives `TUP$getdevaddrs()` its own host Internet address. The programs call the library routine `TUP$inet_addr()` in order to convert the Internet address from a character string to an integer:

```
address = TUP$inet_addr("193.44.5.4");
```

`TUP$getdevaddrs()` returns a pointer to a `dev_addrs` structure:

```
dev_ptr = TUP$getdevaddrs(address);
```

The `dev_addrs` structure contains information about the local system's controllers. This information is used in the `TUP$setup_socket_env()` call. `TUP$getdevaddrs()` also returns the controllers' Internet addresses, which the user program may need later in order to bind sockets.

Calling TUP\$setup_socket_env(): The TUP\$setup_socket_env() routine has the following format:

```
cc = TUP$setup_socket_env(flags, ncontrollers, ControllerListEntry, group);
```

The *flags* parameter indicates whether the call to the controller can request a privileged connection. A privileged connection has a port number less than 1024. (Port numbers from 0 to 1023 are reserved for system use. Many of these reserved port numbers are defined in in.h.)

Note

Any process that requests a privileged connection must be affiliated with the .UBI\$ ACL group, which is created by the System or Network Administrator. If a user who requests a privileged connection is not a member of the .UBI\$ ACL group, an error (EACCESS) is returned.

The *ncontrollers* parameter indicates the number of controllers in *ControllerListEntry*, an array of structures that is defined as follows:

```
typedef struct controller_list_entry { /* Declared in socket.h */
    long   controller_number;
    short  syncs[3];
    short  lcid;
} CONTROLLER_LIST_ENTRY;
```

The *group* parameter, a pointer to a short, indicates whether TUP\$setup_socket_env() should place an already existing event group of ISC synchronizers into the *syncs* field of the *ControllerListEntry* parameter, or should create a new event group. (For information about synchronizers, see the section entitled ISC and Synchronizers, later in this chapter.)

TUP\$setup_socket_env() returns *cc*, which indicates the number of controllers to which a connection was opened.

The call in both the client and server sample programs is

```
cc = TUP$setup_socket_env(0, 1, controller_list, NULL);
```

Both programs give *flags* an argument of 0, requesting a non-privileged connection.

Both the client and the server run on 50 Series systems with one controller, so the *ncontrollers* parameter is set to 1.

The *ControllerListEntry* parameter requires that one of the structure fields, *controller_number*, be filled in before the call. Both programs assign to this field the device address returned from the call to TUP\$getdevaddr():

```
controller_list[0].controller_number = dev_ptr->dev_addr_array[0].dev_addr;
```

When TUP\$setup_socket_env() returns, it fills in the *syncs* field with a list of synchronizers, and fills in the *lcid* field with an ID number for the connection to the controller.

The user programs give *group* a value of `NULL`, to indicate that `TUP$setup_socket_env()` should create a new event group of ISC synchronizers and assign the synchronizers to the *syncs* field of *controller_list*.

A user program can fail to establish a connection to an LHC300 for the following reasons:

- The user program attempted to establish a privileged connection for which it does not have access rights.
- Resources on the host are inadequate. The number of processes wanting to create sockets exceeded the limit (32).
- The controller is not in operation.

If a controller fails, the socket interface sends a signal (`SIGCONTROLLERDOWN`) to indicate a controller failure. (For information about signals, see the section entitled Signals later in this chapter.) The user program must call `TUP$reset_socket_env()` to reestablish broken controller connections.

Before it calls `TUP$reset_socket_env()`, the user program is also responsible for performing a `TUP$close_socket()` for each of the active sockets on the controller that fails, because, as in UNIX, the user program can continue to receive on sockets associated with that connection. A `SIGPIPE` error is signalled only if a user program attempts to send data to a socket on a remote host that is down.

Creating Sockets

Once the socket environment has been established, a socket can be created using the `TUP$socket()` routine. The user program is returned a socket descriptor (an integer) that is used to reference the socket in later socket calls. The format of the `TUP$socket()` routine is

```
s = TUP$socket(af, type, protocol);
```

`TUP$socket()` creates a socket, returning the socket descriptor (*s*) in a specified domain (*af*) and type (*type*), and associates the socket with a protocol (*protocol*). The PRIMOS socket library supports only the Internet domain (`AF_INET`). The `SOCK_STREAM` type supports the TCP protocol, which the client and server programs are using. When a user program does not specify a protocol (enters 0), the socket library selects a protocol associated with the communications type. Therefore, the client and server programs can use the following call to create an Internet socket associated with the TCP protocol:

```
s = TUP$socket (AF_INET, SOCK_STREAM, 0);
```

If the call succeeds, the socket descriptor *s*, an integer that is returned, refers to the socket. The program uses this descriptor to reference the socket in all subsequent calls. If the call fails, the socket library returns a value of `-1`, and specific information is returned in the global variable *errno* (declared in the `stdio.h` file). All socket calls return `ECONTROLLERDOWN` if a controller failure is the cause of the error.

Naming Sockets

The `TUP$socket()` call creates a socket and associates it with a protocol, but it does not name the socket. An application calls `TUP$bind()` to name a socket.

`TUP$bind()` has the following format:

```
cc = TUP$bind(s, name, namelen);
```

s is the socket descriptor that the `TUP$socket()` call returns. *name* is a pointer to a `sockaddr` structure, described in the section entitled Socket Concepts. As that section explained, however, a user program in the Internet domain (`AF_INET`) uses a `sockaddr_in` structure, which includes a port number and an Internet address. Therefore, the client program includes the following code:

```
sock_name.sin_family = AF_INET;
sock_name.sin_port = 0;
sock_name.sin_addr.s_addr = dev_ptr->dev_addr_array[0].internet_address;
cc = TUP$bind(s, (struct sockaddr *)&sock_name, sizeof(sock_name));
```

The program specifies the `AF_INET` (Internet) family. By specifying a port number of 0, the client program asks `TUP$bind()` to assign it a port. (The server program, on the other hand, specifies a predetermined port number of 1101.) The address field is filled in with the value returned from the call to `TUP$getdevadds()`.

The address of the filled-in `sockaddr_in` structure is then cast as a pointer to a `sockaddr` structure for the call to `TUP$bind()`. The *namelen* parameter is given the length in bytes of *name*.

The `TUP$bind()` call enables a process to specify half of the elements that comprise an association (local address, local port). `TUP$connect()` and `TUP$accept()` (described below) are used to complete a socket's association.

Setting and Getting Options

A user program can set and retrieve socket options. The level at which the option resides must be specified so that the selected option can be implemented. The PRIMOS socket implementation supports only the `SOL_SOCKET` level.

For more information about socket options, see the discussions of `TUP$setsockopt()` and `TUP$getsockopt()` in Chapter 9.

Establishing Connections: Client Programs

A client program uses the `TUP$connect()` call to connect a socket to a server's socket. Once the sockets are connected, the client and server processes can transfer data using `TUP$send()` and `TUP$recv()`. Connections are required for connection-based protocols (TCP).

The `TUP$connect()` call can be used with UDP, a connectionless protocol, but no connection is established. With a `SOCK_DGRAM` socket, `TUP$connect()` simply makes a permanent specification of the socket to which data will be sent, so that the socket does not have to be specified each time a packet is sent.

The syntax of `TUP$connect()` is

```
cc = TUP$connect(s, name, namelen);
```

s is the socket descriptor for the local socket. *name* is the name of the remote socket to which the client program is connecting. A program that calls `TUP$connect()` ordinarily declares an `AF_INET` socket name using the more specific `sockaddr_in` structure, and then casts its address to a pointer to a `sockaddr` structure. *namelen* is the length (in bytes) of the socket name.

The client program includes the following statements:

```
rem_addr.sin_family = AF_INET;
rem_addr.sin_port = 1101;
rem_addr.sin_addr.s_addr = address;
cc = TUP$connect(s, (struct sockaddr *)&rem_addr, sizeof(rem_addr));
```

The client must supply the port number and Internet address of the remote server. Symbolic constants that specify port numbers for standard servers (FTP and TELNET, for example) can be found in the `in.h` include file.

Establishing Connections: Server Programs

A server program uses the `TUP$listen()` and `TUP$accept()` calls. `TUP$listen()` prepares the socket to receive connections; `TUP$accept()` waits for connection requests and then completes the connection. Once the sockets are connected, the server and client processes can transfer data using `TUP$send()` and `TUP$recv()`. Connections are required for connection-based protocols (TCP) and are optional for connectionless protocols (UDP).

`TUP$listen()` enables a server to specify a backlog of incoming connections. `TUP$listen()` has the following format:

```
cc = TUP$listen(s, backlog);
```

s is the socket descriptor that `TUP$socket()` returns. *backlog* defines the maximum number of connections that can be queued simultaneously while awaiting a `TUP$accept()` call. (This is *not* the same as the maximum number of connections that can be accepted.) Currently the maximum size of the queue is 5.

The server program uses the following statement to indicate that it is ready to receive requests from clients:

```
cc = TUP$listen(s, 5);
```

The server program calls `TUP$accept()` to accept connections from a remote application. `TUP$accept()` has the following format:

```
ns = TUP$accept(s, addr, addrlen);
```

s is a socket that is listening for connections after a `TUP$listen()`. `TUP$accept()` returns *ns*, a descriptor for the accepted socket. *ns* cannot be used to accept more connections. After the new socket descriptor (*ns*) is returned, the original socket *s* remains open. The new socket is used to send and receive data.

addr is a pointer to a `sockaddr` structure. When `TUP$accept()` returns, *addr* contains the name of the connecting (client) socket. A program that calls `TUP$accept()` ordinarily declares an `AF_INET` socket name using the more specific `sockaddr_in` structure, and then casts its address to a pointer to a `sockaddr` structure.

addrlen is a pointer to the maximum size (in bytes) of the *addr* parameter. Initialize the contents of *addrlen* to indicate the amount of space that *addr* points to; ordinarily, this amount is 16 bytes. When *addr* is returned, *addrlen* points to the actual size of the returned address (in bytes).

The server program includes this call to `TUP$accept()`:

```
rem_addr_len = sizeof(rem_addr);
ns = TUP$accept(s, (struct sockaddr *)&rem_addr, &rem_addr_len);
```

The first of these statements assigns the size in bytes of *rem_addr* to *rem_addr_len*. The second statement passes the socket descriptor and the address of *rem_addr_len* to `TUP$accept()`, which then fills the contents of *rem_addr* with the name of the connecting socket and returns a descriptor for the new socket. To obtain more incoming connections, the program can make repeated `TUP$accept()` calls with the original socket *s* as argument.

Transferring Data

Once a socket has been established, it can be used to receive and transmit data with `TUP$recv()` or `TUP$recvfrom()` and `TUP$send()` or `TUP$sendto()`, respectively. A user program can call `TUP$recv()` or `TUP$send()` only if a socket is connected (or, for a `SOCK_DGRAM` socket, if `TUP$connect()` has permanently specified the destination socket). `TUP$recvfrom()` and `TUP$sendto()` can be called only when a socket is unconnected.

A `SOCK_DGRAM` socket allows sending datagrams to correspondents named in `TUP$sendto()` calls. It is also possible to receive datagrams at such a socket with a `TUP$recvfrom()` call. If a `SOCK_DGRAM` socket wants data to be sent to and received from a specific host only, calling `TUP$connect()` allows the use of `TUP$send()` and `TUP$recv()`.

`TUP$send()` has the following format:

```
cc = TUP$send(s, msg, len, flags);
```

cc, the returned value, is the number of bytes sent. *s* is the socket descriptor returned by `TUP$socket()`. *msg* is a pointer to the message to be transmitted. *len* is the length of the message in bytes. A message delivered by a stream socket may be broken up and delivered in sequence.

The *flags* parameter indicates whether the client is sending normal data (*flags* = 0) or out-of-band data (*flags* = `MSG_OOB`). The PRIMOS socket library supports sending **out-of-band** data on `SOCK_STREAM` sockets. Out-of-band data are high priority messages that the user program wants to deliver instead of previously sent lower-priority messages. Out-of-band data are indicated with a SIGURG signal sent independently of the data. A user program that wants to receive out-of-band data supplies the `MSG_OOB` flag to `TUP$recv()`. The program must establish an on-unit (condition handler) to pick up the SIGURG signal. (See the section entitled Signals, later in this chapter.)

To receive data on a connected socket, a user program calls `TUP$recv()`, which has the following format:

```
cc = TUP$recv(s, buf, len, flags);
```

cc, the returned value, is the number of bytes received. *s* is the socket descriptor returned by `TUP$socket()`. *buf* is a pointer to a message buffer area. When `TUP$recv()` returns, *buf* contains the message received. *len* is the length (in bytes) of *buf*. When *flags* is set to 0 (zero), it indicates that normal data is to be received. *flags* can be set to `MSG_OOB` to receive out-of-band data on `SOCK_STREAM` sockets.

The sample client program first declares and initializes the message buffer *req*:

```
static char req[] = "Send me a message.";
```

The program sends the character string *req* by calling `TUP$send()` as follows:

```
cc = TUP$send(s, req, strlen(req), 0);
```

The server declares the buffer in which it will receive the message as follows:

```
char recvbuff[80];
```

The server program receives the message on the accepted socket (*ns*) with the following call:

```
cc = TUP$recv(ns, recvbuff, sizeof(recvbuff), 0);
```

The server decides what message to send back, assigns the message to the character pointer *cp* and its size to the integer *bufsize*, then sends the information to the client on the accepted socket:

```
cc = TUP$send(ns, cp, bufsize, 0);
```

The client in turn calls `TUP$recv()` to receive the data from the server in its own 80-character message buffer, *buff*:

```
cc = TUP$recv(s, buff, sizeof(buff), 0);
```

The client then displays the contents of the returned buffer on the screen.

Closing Sockets and the Socket Environment

When a user program finishes with a UDP (SOCK_DGRAM) connection, the user can issue a `TUP$shutdown()` to terminate all or part of the connection on that socket. One parameter specifies whether the user wants to be unable to send data, receive data, or both.

Any process that has finished with the socket interface must close each socket using the `TUP$close_socket()` routine and then shut down its controller connections by calling `TUP$close_socket_env()` before quitting. Sockets are not automatically closed when an application terminates.

The client program issues the following calls to close the socket and the socket environment:

```
cc = TUP$close_socket(s);
cc = TUP$close_socket_env();
```

When the client closes the socket environment, the synchronizers that were created by the call to `TUP$setup_socket_env()` are destroyed. If they were added to an already established event group, they are removed from the event group before being destroyed. (For information about synchronizers, see the section entitled *ISC and Synchronizers*, later in this chapter.)

The server program issues a `TUP$close_socket()` call on *ns*, the socket returned by the `TUP$accept()` call. Because the server is a perpetually running program, it never closes its original socket or the socket environment.

An Example of a Datagram Socket Program

In the following example, a client program creates a datagram (connectionless) socket:

```
#include <stdio.h>
#include <in.h>
#include <inet.h>
#include <ioctl.h>
#include <netdb.h>
#include <socket.h>
#include <tcp/ip_error.h>
#include <tcp/ip_time.h>

main()
{

    struct dev_addr *dev_ptr, *TUP$getdevaddrs();
    CONTROLLER_LIST_ENTRY controller_list[2];
    long cc;
    long s;
    long rem_addr_len;
    struct sockaddr_in sock_name;
```

```
long  TUP$setup_socket_env(), TUP$bind(), TUP$socket(),
      TUP$recvfrom(), TUP$sendto(), TUP$close_socket(),
      TUP$close_socket_env();
unsigned long  inet_addr();
struct sockaddr_in  rem_addr;
char  buff[80];
static char  req[] = "Send me a message.";

dev_ptr = TUP$getdevaddrs(0);

controller_list[0].controller_number = dev_ptr->dev_addr_array[0].dev_addr;

cc = TUP$setup_socket_env(0, 1, controller_list, NULL);

s = TUP$socket(AF_INET, SOCK_DGRAM, 0);

sock_name.sin_family = AF_INET;
sock_name.sin_port = 0;
sock_name.sin_addr.s_addr = dev_ptr->dev_addr_array[0].internet_address;

cc = TUP$bind(s, (struct sockaddr *)&sock_name, sizeof(sock_name));

rem_addr.sin_family = AF_INET;
rem_addr.sin_port = 1101;

rem_addr.sin_addr.s_addr = inet_addr("193.44.5.4");

rem_addr_len = sizeof(rem_addr);

cc = TUP$sendto(s, req, strlen(req), 0, &rem_addr, rem_addr_len);

cc = TUP$recvfrom(s, buff, sizeof(buff), 0, &rem_addr, &rem_addr_len);

printf("Returned buffer - \n%s\n", buff);

cc = TUP$close_socket(s);

cc = TUP$close_socket_env();
}
```

To send and receive datagram messages, the client program calls `TUP$sendto()` and `TUP$recvfrom()`. (The connection-oriented client called `TUP$send()` and `TUP$recv()`.)

`TUP$sendto()` has the following format:

```
cc = TUP$sendto(s, msg, len, flags, to, tolen);
```

The *s*, *msg*, *len*, and *flags* parameters are identical to the parameters in `TUP$send()`. *to* is a pointer to the name of the socket to which the datagram is to be sent. *tolen* is the size (in bytes) of *to*.

`TUP$recvfrom()` has the following format:

```
cc = TUP$recvfrom(s, buf, len, flags, from, fromlen);
```

The *s*, *buf*, *len*, and *flags* parameters are identical to the parameters in `TUP$recv()`. *from* returns a pointer to the name of the socket from which the datagram was received. *fromlen* is a pointer to the size (in bytes) of *from*.

Note that the client does not call `TUP$connect()`. Similarly, a server that creates a datagram socket does not call `TUP$listen()` or `TUP$accept()`.

Differences Between PRIMOS and UNIX Socket Support

PRIMOS socket support is based on the standard interface developed at the University of California at Berkeley for the UNIX operating system. The following sections describe the major differences between the 4.3BSD and the PRIMOS implementations.

Sockets and Files

In UNIX implementations of sockets, sockets may be treated as files. In PRIMOS TCP/IP, this is not the case. PRIMOS TCP/IP socket users must call the routines `TUP$send()`, `TUP$sendto()`, `TUP$recv()`, `TUP$recvfrom()`, and `TUP$close_socket()` to send messages, receive messages, and close sockets. Users may *not* call the C library functions `read()`, `write()`, and `close()` for socket operations.

Controllers and Routing

Unlike UNIX implementations, TCP and UDP protocols on a 50 Series host are not part of the operating system but run on one or more LHC300 controllers. This difference affects binding sockets and handling controller failure.

Binding Sockets: 4.3BSD provides the facility of a wildcard address to simplify local address binding. When an application on a host specifies an address as `INADDR_ANY`, the socket interface interprets the address as “any valid address.” (`INADDR_ANY` is a symbolic constant defined in the `socket.h` include file.)

A program on a 50 Series system with one LHC300 controller can use the wildcard facility; sockets that have been bound with a wildcarded local address receive messages directed to the

specified port number on the controller. Wildcarding, however, should not be used on a 50 Series system with two controllers. If a 50 Series system has two controllers, and if the user wishes to receive messages on both controllers, then the server program must create a socket for each controller and bind the proper Internet address to it.

A program that specifies `INADDR_ANY` as part of the remote address argument of the `TUP$send()` call sends data automatically to the first controller on a 50 Series host. (The first controller is the one at `controller_list[0]` in the `TUP$setup_socket_env()` call.) If there are two controllers, the second controller does not receive any messages sent with `INADDR_ANY`.

Controller Failure: PRIMOS must inform a process when a controller fails. PRIMOS thus differs from UNIX, where a process might never know that a controller has failed. Therefore, in the event of a controller failure, the user program must be prepared to clean up its connection to the controller.

If a controller fails, the user program receives a `SIGCONTROLLERDOWN` signal from the socket interface. The user program should then

1. Call `TUP$close_socket()` for each of the active sockets on the controller.
2. Call `TUP$reset_socket_env()` to reestablish broken connections. The socket interface does not send a signal when a controller comes back up. Therefore, the user program should attempt a `TUP$reset_socket_env()` every 30 seconds or so for about five minutes. If the controller does not come up during that time, the user program should close the socket environment and exit the program.

For more information, see the subsection Establishing On-units, later in this chapter.

A `SIGPIPE` condition is signaled only if a user program attempts to send data to or receive data from a socket on a remote host that is down.

Signals

Many errors and other events that arise during execution of socket programs occur asynchronously. Some asynchronous events, such as `SIGPIPE` and `SIGURG`, are signalled by the socket interface to the controller. Others, such as `QUIT$` and `LOGOUT$`, are PRIMOS events. (An asynchronous event is also called a **signal**, **condition**, or **exception**.)

In order to manage these events, the programmer must establish PRIMOS condition handlers (on-units). The PRIMOS subroutine `MKON$P` creates an on-unit by matching a particular user-defined function (the on-unit) to a particular signal. Once `MKON$P` has been called, the function is activated whenever the condition is raised. The user program is responsible for setting up an on-unit for each signal it expects to receive from the socket interface. A call to `TUP$select()` with the exception fields set can identify on which socket the signal occurred.

PRIMOS does not support asynchronous events within one process. This means that the socket interface cannot signal the user until the user calls at least one of the socket library routines.

The PRIMOS socket library can issue the following signals:

<i>Signal</i>	<i>Meaning</i>
SIGURG	Urgent data has been received on a socket.
SIGPIPE	A send() was attempted on a socket that was closed by the remote end.
SIGSOCKETACCEPTED	A socket that was passed to another process has been accepted.
SIGCONTROLLERDOWN	A controller has gone down. For more information, see the next subsection, Establishing On-units.
SIGSOCKETEXCEPTION	An exception has occurred on a socket. The global variable <i>errno</i> (declared in <i>stdio.h</i>) is set to the error that caused the exception. For more information, see the next subsection, Establishing On-units.

The code within an on-unit may make only limited socket subroutine calls. If the on-unit is signalled by one of the five socket library signals listed above, the on-unit may call any of the following routines:

- TUP\$close_socket()
- TUP\$close_socket_env()
- TUP\$reset_socket_env()
- TUP\$select(), if nonblocking I/O has been set by a call to TUP\$ioctl()

If the on-unit is signalled by any other condition, such as the PRIMOS QUIT\$ or ALARM\$ condition, the only socket subroutine the on-unit may call is TUP\$close_socket_env().

Establishing On-units: The following code fragments show how a PRIMOS C server program might set up an on-unit to handle a SIGCONTROLLERDOWN signal.

```
#include <stdio.h>
#include <setjmp.h>

static long  s = -1;    /* make s & ns global and initialize them */
static long  ns = -1;

static long  val;       /* for setjmp() and longjmp() calls */
static jmp_buf  env;

main()
{
    void  chandler();
    fortran  mkon$();
}
```

```
...

mkon$p("SI
...
val = setjmp(env);
...
}

void  chandler(cframe)
int  *cframe;
{
    long  cc, i;
    fortran sleep();

    fprintf(stderr, "Controller down, resetting environment.\n");
    if (s >= 0)
        cc = TUP$close_socket(s);
    if (ns >= 0)
        cc = TUP$close_socket(ns);
    i = 0;
    while ( ( cc = TUP$reset_socket_env()) < 0) && (i < 10) )
    {
        sleep(30);
        i++;
    }
    if (cc < 0)
    {
        printf("Controller still down after 5 minutes, cannot reset.\n");
        TUP$perror("chandler");
        TUP$close_socket_env();
        exit(0);
    }
    else
    {
        printf("Successfully reset environment.\n");
        longjmp(env, val);
    }
}
```

First, the program must declare globally any variables that are to be used by the on-unit. The server program must be prepared to close both the socket *s*, on which it is listening, and the socket *ns*, which is created by the `TUP$accept()` call.

The program also includes the `setjmp.h` standard include file and declares the variables *val* and *env*. The program can now use the `setjmp()` and `longjmp()` functions to return control from the condition-handling function to a particular point in the calling program.

Next, the program declares the condition-handling routine, *chandler*, and the PRIMOS subroutine MKON\$P, which creates an on-unit. At the start of the program, MKON\$P is called with three parameters:

1. A character string containing the name of the condition.
2. A short integer specifying the length of the condition name.
3. The name of the condition-handling function that is to be called when the condition is raised.

The main program calls `setjmp()` at the point to which it wishes to return after *chandler* is invoked.

The condition-handling function should be declared as type `void`. It must have one parameter, which is a pointer to the condition frame on the stack. The sample function displays an error message, closes down any open sockets, and attempts to reset the socket environment.

The socket interface does not send a signal when a controller comes back up. Therefore, the on-unit attempts a `TUP$reset_socket_env()` every 30 seconds or so for about five minutes. If the controller does not come up during that time, the on-unit exits the program. If the controller does come up and the `TUP$reset_socket_env()` succeeds, the on-unit calls `longjmp()` to return to the main program.

If a `SIGSOCKETEXCEPTION` signal is raised, the global variable *errno* (declared in `stdio.h`) is set to the error that caused the exception. A user program can use the contents of the condition frame pointer to find out what socket caused the exception: the *info_ptr* field of the condition frame header structure (described in *Subroutines Reference III: Operating System*) points to the number of the socket.

ISC and Synchronizers

The PRIMOS socket library makes it possible for programs to use the synchronizers associated with the PRIMOS InterServer Communications facility (ISC). ISC, which is similar to the UNIX Inter Process Communications (IPC) facility, is documented in *Subroutines Reference V: Event Synchronization*.

The PRIMOS socket library includes a call, `TUP$setup_socket_env()`, that creates an event group of ISC synchronizers for each LHC300 controller. (Synchronizers are used to indicate for which events the socket code is waiting.) If the user program must use synchronizers in order to use another PRIMOS facility, such as ISC, `TUP$setup_socket_env()` allows the user to add the newly created synchronizers to a preexisting event group. The user program can thus combine the use of sockets and ISC in a single program.

A user program that does not use synchronizers for any other purpose does not need to be aware of synchronizers.

A call to `TUP$setup_socket_env()` gives the user an array of three synchronizers, which have the following names and functions:

- `syncs[0]` (ReadyToReceive) indicates that data is available for reading.
- `syncs[1]` (ReadyToSend) indicates that data is available for writing.
- `syncs[2]` (ExceptionPending) indicates that the system or the remote host has terminated the session.

A user program that wants to use these synchronizers should take the following steps:

1. Wait on the event group for a notice to be posted on a synchronizer.
2. If the synchronizer is socket-related, post the notice again.
3. If you have more than one active socket, make a call to `TUP$select()` to identify the socket on which the event occurred. To identify the event itself, find out which synchronizer in the event group has a notice posted on it.
4. Process the event.

Copying and Sharing Sockets

PRIMOS does not provide a mechanism, such as a `fork()` system call, to copy sockets. `TUP$pass_socket()` passes an open TCP connection from process to process — for example, from a server process to a phantom.

Two different processes cannot share a socket. It is not possible for one process to receive data on a socket and another process to send data on it.

If you want to pass a socket from one process to another, you may use ISC to synchronize the transfer. The processes ordinarily take the following actions:

1. The passing process calls `TUP$pass_socket()`.
2. The passing process uses ISC to identify the socket to the receiving process.
3. The receiving process calls `TUP$receive_passed_socket()`. A `SIGSOCKETACCEPTED` signal notifies the sending process that the socket has been received.

For more information, see the discussions of `TUP$pass_socket()` and `TUP$receive_passed_socket()` in Chapter 9.

Programming Considerations

Include Files for Socket Library Applications

A user program written in the C programming language and incorporating PRIMOS socket routines must include one or more of the following files:

<i>Filename</i>	<i>Description</i>
<code>in.h</code>	Defines and declares Internet constants and structures. These include port numbers, the <code>in_addr</code> structure, and the <code>sockaddr_in</code> structure.
<code>inet.h</code>	Contains external declarations for network library routines.
<code>ioctl.h</code>	Defines the socket options <code>FIONREAD</code> and <code>FIONBIO</code> , changed by <code>TUP\$ioctl()</code> .
<code>netdb.h</code>	Declares structures returned by the network library subroutines.
<code>redefine.h</code>	Contains <code>#define</code> statements that replace the customary UNIX routine names with the corresponding PRIMOS routine names. By using this file, you can recompile a socket program that formerly ran on a UNIX system without having to change the names of the socket routines.
<code>socket.h</code>	Contains socket-related definitions and declarations: types, address families, and options.
<code>stdio.h</code>	Contains PRIMOS C I/O definitions and declarations.
<code>tcp/ip_error.h</code>	Defines socket errors.
<code>tcp/ip_time.h</code>	Declares structures relating to BSD time functions.

All of these include files are in the SYSCOM directory.

Note

The C include files in SYSCOM contain the suffix `.INS.CC`. User programs that include these files may omit this suffix. Files in SYSCOM that lack the `.INS.CC` suffix are not recognized by the C compiler. The `.INS.CC` suffix is optional in other directories. If you copy a header file from SYSCOM to another directory, you may remove the suffix or not, as you wish.

In PRIMOS C, you can specify directories to be searched for include files in a number of different ways. When the C compiler encounters a `#include` directive, it searches for the file in the following manner.

1. If the pathname is delimited by angle brackets (< . >), the compiler goes to step 2.
If the pathname is delimited by double quotation marks (" . ."), the compiler proceeds as follows. If the pathname is a simple filename, the compiler searches the current directory. If the pathname is an absolute pathname (that is, if it begins with a disk partition), the compiler searches that disk partition for the specified path. If the pathname is a full pathname (that is, if it begins with a top-level directory), the compiler searches all the disk partitions for the specified path. If it still cannot find the file, the compiler goes to step 2.
2. The compiler searches the directories specified in command line `-INCLUDE` options, if any. If it cannot find the file in those directories, the compiler goes to step 3.
3. PRIMOS searches the `INCLUDE$` search list and supplies pathnames to the compiler. The compiler then searches these directories. If the compiler cannot find the file in those directories, it goes to step 4.
4. The compiler searches the top-level directory `SYSCOM`. If it cannot find the file, the compiler reports an error.

The PRIMOS search rules facility enables you to establish an `INCLUDE$` search list. An `INCLUDE$` search list is a list of directories that are to be searched for an include file whenever a `#include` directive is processed by the compiler. For complete information about establishing an `INCLUDE$` search list, see the *C User's Guide*.

Calling the Socket Library From Other Languages

You can call the socket library from applications written in languages other than the C programming language. However, special include files are not provided. Therefore, the user program must be responsible for declaring and interpreting correctly all argument types that it passes.

Calling `ER$PRINT` to Display Error Messages

If you wish to display socket error messages at the terminal, but do not want to use the socket routine `TUP$perror()`, you can use the PRIMOS subroutine `ER$PRINT`. The first argument to `ER$PRINT` is a key defined in the file `keys.h`, which you should include in your program. The second argument to `ER$PRINT` should be `tcpip_error_subsystem`, a structure defined in `tcp/ip_error.h`. This argument tells the subroutine to look in `SYSOVL>TUP$_ERROR_TABLE` for the socket error message. The sample server program in Chapter 10 contains a call to `ER$PRINT`. The socket error messages are listed in Appendix G. For more information about `ER$PRINT`, see *Subroutines Reference III: Operating System*.

Compiling an Application

To compile a PRIMOS C program that uses the socket library, use the CC command. You may compile your program in either V mode or IX mode. To compile a program named CLIENT.C in V mode, use the following command:

```
OK, CC CLIENT
```

To compile the same program in IX mode, use the following command:

```
OK, CC CLIENT -32IX
```

These commands produce a binary file called CLIENT.BIN.

Linking an Application

To load the binary version of your program and link it together with the necessary libraries to produce an executable file, use the BIND utility. If you compiled your program in V mode, the following interactive BIND commands produce an executable file:

```
OK, BIND
: LO CLIENT      /* Load the program. */
: LI CCLIB       /* Load the V-mode C library. */
: LI SOCKET      /* Load the socket library. */
: LI             /* Load the system library. */
BIND COMPLETE
: FILE
OK,
```

If you compiled your program in IX mode, use the following BIND commands:

```
OK, BIND
: LO CLIENT      /* Load the program. */
: LI C_LIB       /* Load the IX-mode C library. */
: LI SOCKET      /* Load the socket library. */
: LI             /* Load the system library. */
BIND COMPLETE
: FILE
OK,
```

BIND creates an executable file called CLIENT.RUN.

Running an Application

Once a user program that uses the PRIMOS socket library is written and compiled, it can run under PRIMOS if `LIBRARIES*>SOCKET.RUN` is in the `ENTRY$` search list so that the Executable Program Format (EPF) socket routines are accessible. The TCP/IP installation procedure makes this modification to the systemwide search rules.

To run your program, use the `RESUME` command. You may omit the `.RUN` suffix.

```
OK, RESUME CLIENT
```

Compatibility

A network application written in C and compiled in V mode on a 50 Series system can run on almost any 50 Series system. However, programs compiled in IX mode do not run on older machines. For more information, see the *C User's Guide*.

Restrictions and Limitations

This section describes the restrictions and limitations that apply to Release 2.0 of the PRIMOS socket library:

- The PRIMOS socket library supports only
 - TCP and UDP protocols
 - The `AF_INET` domain
- Options can be set only at the `SOL_SOCKET` level. The `IPPROTO_TCP` and `IPPROTO_IP` socket levels are not supported.
- A process that calls the PRIMOS socket library can use a maximum of 32 sockets simultaneously.
- A maximum of 32 processes can use the socket interface to communicate with each LHC300 controller. PRIMOS TCP/IP supports a maximum of two controllers per system. Therefore, a maximum of 64 processes on a system can use the socket interface simultaneously.

A process that has created one or more sockets on two LHC300s is counted against the 32 users of each controller. This reduces the total number of possible user processes per system. The user TELNET and user FTP programs are implemented with the socket interface. Therefore, the number of active TELNET and FTP users can reduce the number of user-written application processes permitted to call the interface.

Socket Subroutine Calls

This chapter describes the subroutines that comprise the PRIMOS socket library and the subroutines required to support the socket environment on a 50 Series system. Read Chapter 8, Introduction to the PRIMOS Socket Library, before you read this chapter.

Subroutines fall logically into the following groups:

- Socket environment — Subroutines required to initialize the socket environment on a 50 Series system
- Socket — Socket subroutines supported under PRIMOS on a 50 Series system
- Network library — Subroutines that perform functions such as mapping host names to Internet addresses, mapping Internet addresses to network numbers, and converting addresses from character strings to integers and back again

Subroutines are described in a standard format in alphabetical order within each group.

Summary of Routines

Table 9-1 lists in alphabetical order the socket environment subroutines described in this chapter.

Table 9-1
Socket Environment Subroutine Calls

<i>Subroutine</i>	<i>Description</i>
TUP\$close_socket_env()	Closes the socket environment.
TUP\$pass_socket()	Passes a socket to another process.
TUP\$receive_passed_socket()	Accepts a socket from another process.
TUP\$reset_socket_env()	Resets the socket environment.
TUP\$setup_socket_env()	Initializes the socket environment.

Table 9-2 lists in alphabetical order the PRIMOS socket subroutines described in this chapter.

Table 9-2
PRIMOS Socket Subroutine Calls

<i>Subroutine</i>	<i>Description</i>
TUP\$accept()	Accepts an incoming connection request for a server's socket.
TUP\$bind()	Associates a name with a socket.
TUP\$close_socket()	Removes a socket entry.
TUP\$connect()	Initiates a client's connection to a server's socket.
TUP\$getpeername()	Returns the name of a socket's peer (the socket at the other end of the connection).
TUP\$getsockname()	Returns the current name for a socket.
TUP\$getsockopt()	Retrieves the values assigned to socket options.
TUP\$ioctl()	Performs control operations.
TUP\$listen()	Listens on a bound socket for connection requests from client processes.
TUP\$perror()	Displays an error message at the terminal.
TUP\$recv()	Receives messages from a connected socket.
TUP\$recvfrom()	Receives messages from an unconnected socket.
TUP\$select()	Monitors multiple I/O requests among multiple sockets.
TUP\$send()	Sends a message to a connected socket.
TUP\$sendto()	Sends a message to an unconnected socket.
TUP\$setsockopt()	Sets socket options.
TUP\$shutdown()	Shuts down all or part of a full-duplex connection on a socket.
TUP\$socket()	Creates a socket.

Table 9-3 lists in alphabetical order the network library subroutines described in this chapter.

Table 9-3
Network Library Subroutine Calls

<i>Subroutine</i>	<i>Description</i>
TUP\$getalladdrs()	Returns all the Internet addresses of a host when given any of the alternate names of the host.
TUP\$getbroadcastaddr()	Returns the broadcast address for a controller.
TUP\$getdevaddrs()	Returns the device addresses of LHC300 controllers.
TUP\$gethostbyaddr()	Returns information about the host when given an Internet address of the host.
TUP\$gethostbyname()	Returns information about the host when given an official name or alias name of the host.
TUP\$gethostname()	Returns the name of the local host.
TUP\$inet_addr()	Returns an unsigned long integer when given an Internet address in dot notation.
TUP\$inet_lnaof()	Returns a local address when given an Internet address specified as a structure.
TUP\$inet_makeaddr()	Returns an Internet address when given the network number and the local address specified as long integers.
TUP\$inet_netof()	Returns the network number when given an Internet address specified as a structure.
TUP\$inet_network()	Returns an unsigned long integer when given a network number in dot notation.
TUP\$inet_ntoa()	Returns an Internet address as a character string when given an Internet address specified as a structure.

Subroutine Descriptions

Each description of a subroutine begins with a summary of what the subroutine call does. It then includes the following sections:

- **Call Syntax.** The format of a subroutine call and parameter declarations using the elements of the C programming language.
- **Parameters.** Information about the arguments that the subroutine expects and the values that it returns. The argument values of parameters designated INPUT are not changed by the subroutine. The argument values of parameters designated OUTPUT

are changed by the subroutine. You do not have to initialize the arguments of parameters labeled OUTPUT before you call the subroutine. The argument values of parameters labeled INPUT/OUTPUT may be changed by the subroutine.

- **Errors.** Information about any error codes that the subroutine can return and that describe why a subroutine call can fail. See Appendix G for a complete list of socket library error messages.

Include Files

Any user program that calls the socket library subroutines must make some of the following `#include` requests to the C preprocessor:

```
#include <stdio.h>
#include <in.h>
#include <inet.h>
#include <ioctl.h>
#include <netdb.h>
#include <socket.h>
#include <tcp/ip_error.h>
#include <tcp/ip_time.h>
```

The Call Syntax section of each subroutine description indicates which include files are required for that subroutine.

Socket Environment Subroutine Calls

The subroutine calls described in this section support the PRIMOS socket library on a 50 Series system.

TUP\$close_socket_env()

TUP\$close_socket_env() closes down controller connections that are used in the socket environment.

When a process that is utilizing the socket interface has closed all sockets and is about to exit, it must call TUP\$close_socket_env() to release the system resources that TUP\$setup_socket_env() has allocated.

WARNING

If the process does not call TUP\$close_socket_env(),

- Any subsequent calls to TUP\$setup_socket_env() produce unpredictable results.
 - The third call to TUP\$setup_socket_env() causes the global variable *errno* (declared in *stdio.h*) to be set to EUBIERR.
 - The system resources remain allocated until the process either logs out or issues an INITIALIZE_COMMAND_ENVIRONMENT (ICE) command.
-

TUP\$close_socket_env() destroys the synchronizers established by the TUP\$setup_socket_env() call that initialized the socket environment. If the synchronizers were added to a previously established event group of ISC synchronizers, TUP\$close_socket_environment removes them from the event group and then destroys them. The contents of the *ControllerListEntry[]*.*syncs* array returned by TUP\$setup_socket_env() are invalidated. For more information, see the discussion of TUP\$setup_socket_env().

Call Syntax

```
#include <tcp/ip_error.h>
```

```
cc = TUP$close_socket_env();
```

```
long cc;
```

Parameters

cc

OUTPUT. If the call succeeds, a 0 is returned. If the call fails, a value of -1 is returned and the global variable *errno* (declared in *stdio.h*) provides more specific error information.

Errors

The following error code describes why a `TUP$close_socket_env()` call can fail:

<i>Error Code</i>	<i>Description</i>
ESYSERR	A system error has occurred. Contact your System or Network Administrator.
EUBIERR	The connection to the controller has not been established or has been interrupted.

TUP\$pass_socket()

TUP\$pass_socket() passes a socket from the calling process to another process.

Servers that spawn separate processes to handle specific connections must be able to pass a socket to another process. In UNIX, this happens automatically whenever a child process is forked. Forking, as it is defined in UNIX, cannot be done under PRIMOS. Therefore, the user program must call TUP\$pass_socket().

After a socket is passed off, the process that issues the TUP\$receive_passed_socket() can send data, receive data, or perform any other socket operations on that socket. The original owner of the socket can no longer issue any socket calls on that socket. When the receiving process accepts a socket, the process giving up the socket is informed by the SIGSOCKETACCEPTED signal. Therefore, a process that expects to pass a socket to another process must set up an on-unit (condition handler) to handle this event.

Call Syntax

```
#include <tcp/ip_error.h>
```

```
state = TUP$pass_socket(s, uid);
```

```
long s, uid;
struct {
    short length;
    char string[128];
} *state;
```

Parameters

s

INPUT. Socket descriptor (returned by a TUP\$socket() call) for the socket that is to be passed to another process. When a user program issues a TUP\$pass_socket(), no data can be received or sent on the socket specified by *s* until a TUP\$receive_passed_socket() is issued with the same *uid*.

uid

INPUT. Unique ID that TUP\$pass_socket() uses to generate a string for the returned value. *uid* is a security mechanism; it ensures that the wrong process does not intercept the socket. *uid* should be a random number chosen each time TUP\$pass_socket() is used.

state

OUTPUT. Returned value. A pointer to a structure that consists of the string length followed by a character array with the actual string. The string, which may be a maximum of 128 characters long, contains an encoded version of *uid* as well as encoded information about the socket and its environment. If an error is detected, a null pointer is returned and the global variable *errno* (declared in *stdio.h*) provides more specific error information. If the call succeeds, the process that calls `TUP$pass_socket()` must then communicate the *state* to the process that will issue the `TUP$receive_passed_socket()` call. The process can communicate using any means (for example, ISC).

Errors

The following error codes describe why a `TUP$pass_socket()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONTROLLERDOWN	The controller is not in operation.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$receive_passed_socket()

TUP\$receive_passed_socket() accepts a socket from another process. After one process has issued a TUP\$pass_socket() with a unique ID, it sends the child process a message consisting of the contents of the *state* pointer. The child then issues a TUP\$receive_passed_socket() call to receive the socket, with a pointer to the message as the argument to the call.

Call Syntax

```
#include <tcp/ip_error.h>

s = TUP$receive_passed_socket(state) ;

long s;
struct {
    short length;
    char string[128];
} *state;
```

Parameters

state

INPUT. A pointer to a structure that consists of the string length followed by a character array with the actual string. See the description of TUP\$pass_socket() for information about the contents of the string.

s

OUTPUT. The socket number of the passed socket. If the call fails, a value of -1 is returned and the global variable *errno* (declared in *stdio.h*) provides more specific error information.

Errors

The following error codes describe why a TUP\$receive_passed_socket() call can fail:

<i>Error Code</i>	<i>Description</i>
ECONTROLLERDOWN	The controller is not in operation.
ENOBUFS	The system did not have sufficient resources to perform the operation.
ENOTSOCK	The string pointed to by <i>state</i> does not match the string returned by a TUP\$pass_socket() call, and the socket is still waiting to be received.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$reset_socket_env()

TUP\$reset_socket_env() reestablishes connections to controllers that have gone down.

The user program must call TUP\$reset_socket_env() after it receives a SIGCONTROLLERDOWN signal from the socket interface. SIGCONTROLLERDOWN indicates a controller failure. The user program must call TUP\$reset_socket_env() to reestablish broken connections.

Before it calls TUP\$reset_socket_env(), the user program is also responsible for performing a TUP\$close_socket() for each of the active sockets on the controller that fails, because, as in UNIX, the user program can continue to receive data on sockets associated with that connection. A SIGPIPE error is signaled only if a user program attempts to send data to or receive data from a socket on a remote host that is down.

Call Syntax

```
#include <socket.h>
#include <tcp/ip_error.h>

cc = TUP$reset_socket_env(flags, ncontrollers, ControllerListEntry);

long flags, ncontrollers, cc;
CONTROLLER_LIST_ENTRY ControllerListEntry[];

typedef struct controller_list_entry { /* Declared in socket.h */
    long controller_number;
    short syncs[3];
    short lcid;
} CONTROLLER_LIST_ENTRY;
```

Parameters

flags

INPUT. Request flags for this connection. *flags* can be set to either 0 or SO_PRIVILEGED, a symbolic constant defined in the socket.h file. SO_PRIVILEGED indicates that the open call to the controller is allowed to request privileged connections, as in the case of the FTP server. A privileged connection is one with a port number less than 1024. A server program cannot set *flags* to SO_PRIVILEGED unless the server is a member of the .UBI\$ ACL group.

ncontrollers

INPUT. The number of controllers in the *ControllerListEntry* array.

ControllerListEntry

INPUT/OUTPUT. An array of structures that is a list of controllers for which connections are to be established.

controller_number

INPUT. A structure member that specifies the device address of the controller.

syncs

OUTPUT. A returned structure member that consists of an array, which is an event group of synchronizers. The *ControllerListEntry[]*.*syncs[]* field is filled in with the synchronizer identifiers for each valid controller. If a controller is not present or is incorrectly specified, the array elements are set equal to -1. For more information about synchronizers, see the subsection ISC and Synchronizers in Chapter 8.

The following information is returned for each controller:

<i>ControllerListEntry[]</i> . <i>syncs[0]</i>	Synchronizer for input data
<i>ControllerListEntry[]</i> . <i>syncs[1]</i>	Synchronizer for output data
<i>ControllerListEntry[]</i> . <i>syncs[2]</i>	Synchronizer for termination

lcid

OUTPUT. A returned structure member that is a controller connection ID. The *ControllerListEntry[]*.*lcid* field is filled in with the corresponding information for each valid controller. If a controller is not present or is incorrectly specified, this field is set equal to -1.

The following information is returned for each controller:

<i>ControllerListEntry[]</i> . <i>lcid</i>	ID for the connection
--	-----------------------

cc

OUTPUT. If the call succeeds,

- The returned value is the number of controllers that were opened successfully (0 if none).
- The *ControllerListEntry[]*.*syncs* and *ControllerListEntry[]*.*lcid* fields are filled in with the corresponding information for each connection that was reestablished. If a connection could not be established, these fields are set to -1.

If the call fails, a value of -1 is returned and the global variable *errno* (declared in *stdio.h*) provides specific error information.

Errors

The following error codes describe why a `TUP$reset_socket_env()` call can fail:

<i>Error Code</i>	<i>Description</i>
EACCESS	The user requested a privileged connection, but the current user has inadequate permission to access it.
ENOBUFS	The system did not have sufficient resources to perform the operation.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$setup_socket_env()

TUP\$setup_socket_env() initializes the user socket environment on a 50 Series system.

Before the user program can access the socket library, the user program must initialize the socket environment. Setting up the environment entails establishing a connection for each of the controllers.

Call Syntax

```
#include <socket.h>
#include <tcp/ip_error.h>

cc = TUP$setup_socket_env(flags, ncontrollers, ControllerListEntry, group);

long flags, ncontrollers, cc;
CONTROLLER_LIST_ENTRY ControllerListEntry[];
short *group;

typedef struct controller_list_entry { /* Declared in socket.h */
    long controller_number;
    short syncs[3];
    short lcid;
} CONTROLLER_LIST_ENTRY;
```

Parameters

flags

INPUT. Request flags for this connection. *flags* can be set either to 0 or to SO_PRIVILEGED, a symbolic constant defined in the socket.h file. SO_PRIVILEGED indicates that the open call to the controller is allowed to request privileged connections, as in the case of the FTP server. A privileged connection is one with a port number less than 1024. A server program cannot set *flags* to SO_PRIVILEGED unless the server is a member of the .UBI\$ ACL group.

ncontrollers

INPUT. Number of controllers in the *ControllerListEntry* array.

ControllerListEntry

INPUT/OUTPUT. An array of structures that is a list of controllers for which connections are to be established.

controller_number

INPUT. A structure member that specifies the device address of the controller. The user can obtain this address by calling `TUP$getdevaddr()`.

syncs

OUTPUT. A returned structure member that consists of an array, which is an event group of synchronizers that `TUP$setup_socket_env()` has created. The *ControllerListEntry[]*.*syncs[]* field is filled in with the synchronizer identifiers for each valid controller. If a controller is not present or is incorrectly specified, the array elements are set equal to -1. For more information about synchronizers, see the subsection ISC and Synchronizers in Chapter 8.

The following information is returned for each controller:

<i>ControllerListEntry[]</i> . <i>syncs</i> [0]	Synchronizer for input data
<i>ControllerListEntry[]</i> . <i>syncs</i> [1]	Synchronizer for output data
<i>ControllerListEntry[]</i> . <i>syncs</i> [2]	Synchronizer for termination

lcid

OUTPUT. A returned structure member that is a controller connection ID. The *ControllerListEntry[]*.*lcid* field is filled in with the corresponding information for each valid controller. If a controller is not present or is incorrectly specified, this field is set equal to -1.

The following information is returned for each controller:

<i>ControllerListEntry[]</i> . <i>lcid</i>	ID for the connection
--	-----------------------

group

INPUT/OUTPUT. A pointer to a short integer. If the pointer is not NULL, `TUP$setup_socket_env()` adds the synchronizers to the event group of ISC synchronizers that has the number specified by the contents of the pointer. An ESYSERR error message is sent if the user specifies an invalid event group number. If the *group* pointer is NULL, `TUP$setup_socket_env()` creates a new event group. The number of the newly created event group can be retrieved by a call to one of the synchronizer subroutines described in *Subroutines Reference V: Event Synchronization*.

cc

OUTPUT. If the call succeeds,

- The returned value is the number of controllers that were opened successfully (0 if none).
- The *ControllerListEntry[]*.*syncs* and *ControllerListEntry[]*.*lcid* fields are filled in with the corresponding information for each connection that was established. If a connection could not be established, these fields are set to -1.

If the call fails, a value of `-1` is returned and the global variable *errno* (declared in `stdio.h`) returns specific error information.

Errors

The following error codes describe why a `TUP$setup_socket_env()` call can fail:

<i>Error Code</i>	<i>Description</i>
EACCESS	The user requested a privileged connection, but the current user has inadequate permission to access it.
EFAULT	The <i>ControllerListEntry</i> pointer is invalid.
ENOBUFS	The system did not have sufficient resources to perform the operation.
ESYSERR	A system error has occurred. An invalid event group number may have been specified. If this is not the source of the problem, contact your System or Network Administrator.
EUBIERR	The connection to the controller has not been established or has been interrupted.

PRIMOS Socket Subroutine Calls

The next sections describe the PRIMOS socket subroutine calls comparable to 4.3BSD calls.

TUP\$accept()

TUP\$accept() accepts an incoming connection request for a socket from a client process. TUP\$accept() applies only to connection-based protocols (TCP). Therefore, it may be used only with SOCK_STREAM sockets.

A program calls TUP\$accept() after a call to TUP\$listen(). TUP\$accept() performs the following operations:

1. It extracts the first connection on a queue of pending connections.
2. It creates a new socket that has the same properties as the socket created with TUP\$socket().
3. It returns a new socket descriptor for the socket.

The new socket cannot accept more connections. The original socket remains open.

If no pending connections are on the queue and a socket is marked as blocking (the default), TUP\$accept() blocks the calling program. That is, it does not return until a connection is present or a signal to a process interrupts a system call.

If the socket is marked nonblocking and no pending connections are on the queue, TUP\$accept() returns an error. (Use the TUP\$ioctl() routine to mark a socket nonblocking.)

Call Syntax

```
#include <socket.h>
#include <in.h>
#include <tcp/ip_error.h>

ns = TUP$accept(s, addr, addrlen);

long s, ns;
struct sockaddr *addr;
long *addrlen;

struct sockaddr {          /* Declared in socket.h */
    short sa_family;
    char sa_data[14];
};
```

Parameters

s

INPUT. Socket descriptor returned by TUP\$socket() call. The argument *s* is a socket that has been created with TUP\$socket(), has optionally been bound to an address with TUP\$bind(), and is listening for connections after a TUP\$listen(). The socket type of *s* must be SOCK_STREAM. After a new socket descriptor (*ns*) is returned, *s* remains in existence. It can be used to accept another connection.

addr

INPUT/OUTPUT. Pointer to sockaddr structure, which when returned is the name of the connecting (client) socket. A program that calls TUP\$accept() ordinarily declares an AF_INET socket name using the more specific sockaddr_in structure, and then casts its address to a pointer to a sockaddr structure. If the TUP\$accept() call fails, the structure is not updated. The sockaddr_in structure is declared as follows:

```
struct sockaddr_in {    /* Declared in in.h */
    short              sin_family;
    unsigned short     sin_port;
    struct in_addr      sin_addr;
    char               sin_zero[8];
};

struct in_addr {        /* Declared in in.h */
    unsigned long      s_addr;
};
```

sa_family

INPUT. A structure member that identifies the address format. The corresponding field in the sockaddr_in structure is *sin_family*. The PRIMOS socket library supports only the AF_INET family.

sa_data

INPUT. A structure member that is a 14-byte character array containing the actual data value. The corresponding fields in the sockaddr_in structure are *sin_port*, *sin_addr*, and *sin_zero*.

addrlen

INPUT/OUTPUT. Pointer to maximum size (in bytes) of *addr*. Initialize the contents of *addrlen* to indicate the amount of space pointed to by *addr* (ordinarily 16 bytes). When *addr* is returned, *addrlen* points to the actual size of the returned address (in bytes). If the call fails, the contents of *addrlen* are not updated.

ns

OUTPUT. Descriptor for the accepted socket. If the call succeeds, it returns a non-negative integer. If an error occurs, the call returns a value of -1 and the global variable *errno* (declared in *stdio.h*) provides specific error information. *ns* cannot be used to accept more connections. The original socket (*s*) remains capable of accepting connections.

Errors

The following error codes describe why a `TUP$accept()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONTROLLERDOWN	The controller is not in operation.
ENOBUFS	The system did not have sufficient resources to perform the operation.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
EOPNOTSUPP	The referenced socket type is not <code>SOCK_STREAM</code> .
ESYSERR	A system error has occurred. Contact your System or Network Administrator.
EWOULDBLOCK	The socket is marked nonblocking and no connections exist to be accepted.

TUP\$bind()

TUP\$bind() assigns a name to an unnamed socket. The name remains bound to the socket until the socket is closed by a call to TUP\$close_socket(). (For information on socket names, see Chapter 8.)

When a socket is created with TUP\$socket(), the socket exists in a name space (also called an address family or domain), but no name is assigned. If the socket has no name, the socket interface library on the 50 Series system and the interface process on the LHC300 controller cannot communicate with each other. Consequently, the socket cannot receive messages.

All addresses belong to one or more address families (domains), which define their format and interpretation. In the AF_INET domain, the bound name contains the Internet address of the network interface and the port number on which the process expects data.

The PRIMOS socket library routines use the generic sockaddr structure to describe a socket name. Programs that call these routines ordinarily define an AF_INET socket name using the more specific sockaddr_in structure, and then cast its address to a pointer to a sockaddr structure. (See Chapter 8.)

Note

In general, a user program expected to run on a 50 Series system with more than one LHC300 running TCP/IP should explicitly create a socket for each controller and bind the proper Internet address to each. Each controller has a different Internet address.

Call Syntax

```
#include <in.h>
#include <socket.h>
#include <tcp/ip_error.h>

cc = TUP$bind(s, name, namelen);

long s, namelen, cc;
struct sockaddr *name;

struct sockaddr {          /* Declared in socket.h */
    short sa_family;
    char sa_data[14];
};
```

Parameters

s

INPUT. Socket descriptor returned by a TUP\$socket() call.

name

INPUT. Pointer to sockaddr structure that is an address to be assigned to *s*. The user program ordinarily declares a sockaddr_in structure:

```
struct sockaddr_in {                /* Declared in in.h */
    short          sin_family;
    unsigned short sin_port;
    struct in_addr  sin_addr;
    char           sin_zero[8];
};

struct in_addr {                    /* Declared in in.h */
    unsigned long   s_addr;
};
```

The *sin_port* field of this structure may specify a port number of 0 (zero) to have the system select a port to use. If the 50 Series system has only one LHC300 controller, the structure may contain both a port of 0 (zero) and an INADDR_ANY address in the *sin_addr* field. In this case, TUP\$bind() fills in both a port and an address for the socket. If the 50 Series system has more than one controller, you must specify the controller address.

sa_family

INPUT. A structure member that identifies the address format. The corresponding field in the sockaddr_in structure is *sin_family*. The PRIMOS socket library supports only the AF_INET family.

sa_data

INPUT. A structure member that is a 14-byte character array containing the actual data value. The corresponding fields in the sockaddr_in structure are *sin_port*, *sin_addr*, and *sin_zero*.

namelen

INPUT. Length (in bytes) of *name*.

cc

OUTPUT. Return status. If the bind is successful, 0 is returned. If the call fails, a value of -1 is returned and the global variable *errno* (declared in stdio.h) provides specific error information.

Errors

The following error codes describe why a TUP\$bind() call can fail:

<i>Error Code</i>	<i>Description</i>
EACCESS	The requested address is protected and the user program has inadequate permission to access it. This only happens when the user program tries to use a port with a value less than 1024, because these ports are reserved for system use. TUP\$setup_socket_env() determines the privilege of a user program.
EADDRINUSE	The specified address is already in use and the SO_REUSEADDR flag (socket option) is not set. See the discussion of TUP\$setsockopt().
EADDRNOTAVAIL	The specified address is not available from the local machine.
EBADF	<i>s</i> falls outside the range of valid socket descriptors.
ECONTROLLERDOWN	The controller is not in operation.
EINVAL	The socket is already bound to an address.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$close_socket()

A user program that finishes with a socket or connection can call TUP\$close_socket() to remove the socket entry. Sockets are not closed automatically when a program terminates.

If data are associated with a socket that promises reliable delivery (that is, a SOCK_STREAM socket), the system attempts to transfer the data when a TUP\$close_socket() occurs. However, if data are not delivered after a long period of time, the data are discarded.

Note

For connectionless protocols (SOCK_DGRAM type sockets), TUP\$close_socket() applies locally only. If the program uses a connection-oriented protocol, and if it does not set the SO_LINGER option, a return from TUP\$close_socket() does not guarantee that the connection has gone away.

Call Syntax

```
#include <tcp/ip_error.h>
```

```
cc = TUP$close_socket(s);
```

```
long s, cc;
```

Parameters

s

INPUT. Socket descriptor returned by TUP\$socket().

cc

OUTPUT. Return status. If the call succeeds, 0 is returned. If the call fails, a value of -1 is returned and the global variable *errno* (declared in *stdio.h*) provides specific error information.

Errors

The following error codes describe why a TUP\$close_socket() call can fail:

Error Code

Description

EBADF

s is not a valid socket descriptor.

ECONTROLLERDOWN

The controller is not in operation.

ENOTSOCK

The descriptor *s* is not an active socket.

ESYSERR

A system error has occurred. Contact your System or Network Administrator.

TUP\$connect()

A client program that uses `SOCK_STREAM` sockets calls `TUP$connect()` to initiate a connection to a server's socket.

If the program uses `SOCK_DGRAM` sockets, `TUP$connect()` does not actually make a connection. Instead, it permanently specifies the socket's destination address so that the user can use `TUP$send()` and `TUP$recv()` to transfer data. `TUP$connect()` can, therefore, be useful if all datagrams are to be sent to the same destination.

If the client's socket is unbound when it issues a `TUP$connect()`, a name is automatically selected and bound to the socket. Calling `TUP$connect()` on an unbound socket is equivalent to calling `TUP$bind()` with a name consisting of `INADDR_ANY` and a port of 0 (zero). Therefore, a client program on a 50 Series system that has only one LHC300 controller does not have to issue a `TUP$bind()` call in order to make a connection from a socket. If the client omits the `TUP$bind()` call, it may call `TUP$getsockname()` in order to find out the socket name. Refer to the section entitled `TUP$bind()` for information on binding a socket.

An error is returned when the call is unsuccessful. The name that is bound to the socket during the connection attempt is retained.

Call Syntax

```
#include <in.h>
#include <socket.h>
#include <tcp/ip_error.h>

cc = TUP$connect(s, name, namelen);

long s, namelen, cc;
struct sockaddr *name;

struct sockaddr {          /* Declared in socket.h */
    short sa_family;
    char sa_data[14];
};
```

Parameters

s

INPUT. Socket descriptor returned by `TUP$socket()`. If *s* is of type `SOCK_DGRAM`, `TUP$connect()` permanently specifies the peer to which datagrams are to be sent. If *s* is of type `SOCK_STREAM`, `TUP$connect()` attempts to make a connection to another socket. For both socket types the *name* parameter, which is associated with an address in the `AF_INET` domain, specifies the other socket.

name

INPUT. Pointer to `sockaddr` structure that specifies the name of the destination socket. A program that calls `TUP$connect()` ordinarily declares an `AF_INET` socket name using the more specific `sockaddr_in` structure, and then casts its address to a pointer to a `sockaddr` structure. The `sockaddr_in` structure is declared as follows:

```
struct sockaddr_in {    /* Declared in in.h */
    short      sin_family;
    unsigned short sin_port;
    struct in_addr sin_addr;
    char       sin_zero[8];
};

struct in_addr {    /* Declared in in.h */
    unsigned long  s_addr;
};
```

sa_family

INPUT. A structure member that identifies the address format. The corresponding field in the `sockaddr_in` structure is `sin_family`. The PRIMOS socket library supports only the `AF_INET` family.

sa_data

INPUT. A structure member that is a 14-byte character array containing the actual data value. The corresponding fields in the `sockaddr_in` structure are `sin_port`, `sin_addr`, and `sin_zero`.

namelen

INPUT. Length (in bytes) of the socket *name*. Initialize *namelen* to indicate the amount of space pointed to by *name* (ordinarily 16 bytes).

cc

OUTPUT. Return status. If the call succeeds, *cc* is 0. If the call fails, a value of -1 is returned and a more specific error code is stored in the global variable *errno* (declared in `stdio.h`). A successful return from `TUP$connect()` does not necessarily mean that a connection has been established; to trap connection errors, set up an on-unit for the `SIGSOCKETEXCEPTION` condition.

Errors

The following error codes describe why a `TUP$connect()` call can fail:

<i>Error Code</i>	<i>Description</i>
EADDRINUSE	The address is already in use.

EAFNOSUPPORT	Addresses in the specified address family cannot be used with this socket.
EBADF	<i>s</i> is not a valid socket descriptor.
ECONNREFUSED	The attempt to connect has been rejected. This error appears only if the program has established an on-unit for SIGSOCKETEXCEPTION.
ECONTROLLERDOWN	The controller is not in operation.
EINVAL	The value specified for <i>namelen</i> is invalid.
EISCONN	The socket is already connected.
ENETUNREACH	The network is not reachable from this host.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.
ETIMEDOUT	The attempt to establish a connection timed out without establishing a connection. This error occurs only if I/O is blocking (the default).
EWouldBLOCK	The socket is nonblocking and the connection cannot be completed immediately. It is possible to select the socket for sending to determine when the connection was established. See TUP\$select().

TUP\$getpeername()

TUP\$getpeername() returns the name of the peer connected to a SOCK_STREAM socket. (The *peer* is the socket at the other end of the TCP connection.)

TUP\$getpeername() does not apply to SOCK_DGRAM sockets.

Call Syntax

```
#include <in.h>
#include <socket.h>
#include <tcp/ip_error.h>

cc = TUP$getpeername(s, name, namelen);

long s, cc;
struct sockaddr *name;
long *namelen;

struct sockaddr {          /* Declared in socket.h */
    short sa_family;
    char sa_data[14];
};
```

Parameters

s

INPUT. Socket descriptor returned by TUP\$socket() call.

name

OUTPUT. Pointer to sockaddr structure, which names the peer connected to the socket *s*. A program that calls TUP\$getpeername() ordinarily declares an AF_INET socket name using the more specific sockaddr_in structure, and then casts its address to a pointer to a sockaddr structure. If the TUP\$getpeername() call fails, the structure is not updated. The sockaddr_in structure is declared as follows:

```
struct sockaddr_in {      /* Declared in in.h */
    short          sin_family;
    unsigned short sin_port;
    struct in_addr sin_addr;
    char           sin_zero[8];
};

struct in_addr {          /* Declared in in.h */
    unsigned long  s_addr;
};
```

sa_family

OUTPUT. A structure member that identifies the address format. The corresponding field in the `sockaddr_in` structure is *sin_family*. The PRIMOS socket library supports only the `AF_INET` family.

sa_data

OUTPUT. A structure member that is a 14-byte character array containing the actual data value. The corresponding fields in the `sockaddr_in` structure are *sin_port*, *sin_addr*, and *sin_zero*.

namelen

INPUT/OUTPUT. Pointer to maximum size (in bytes) of the `sockaddr_in` structure. Initialize the contents of *namelen* to indicate the amount of space pointed to by *name* (ordinarily 16 bytes). When *namelen* is returned, it contains the actual size of the name returned (in bytes). If the call fails, the contents of *namelen* are not updated.

cc

OUTPUT. Return status. A value of 0 is returned if the call succeeds. If the call fails, a value of -1 is returned and the global variable *errno* (declared in `stdio.h`) returns specific error information.

Errors

The following error codes describe why a `TUP$getpeername()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONTROLLERDOWN	The controller is not in operation.
ENOBUFS	The system did not have sufficient resources to perform the operation.
ENOTCONN	The specified socket is not connected.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$getsockname()

TUP\$getsockname() returns the current name for the specified socket.

Call Syntax

```
#include <in.h>
#include <socket.h>
#include <tcp/ip_error.h>

cc = TUP$getsockname(s, name, namelen);

long s, cc;
struct sockaddr *name;
long *namelen;

struct sockaddr {          /* Declared in socket.h */
    short sa_family;
    char sa_data[14];
};
```

Parameters

s

INPUT. Socket descriptor returned by TUP\$socket() call.

name

OUTPUT. Pointer to the sockaddr structure in which the socket name is to be returned. A program that calls TUP\$getsockname() ordinarily declares an AF_INET socket name using the more specific sockaddr_in structure, and then casts its address to a pointer to a sockaddr structure. If the TUP\$getsockname() call fails, the structure is not updated. The sockaddr_in structure is declared as follows:

```
struct sockaddr_in {      /* Declared in in.h */
    short      sin_family;
    unsigned short sin_port;
    struct in_addr sin_addr;
    char       sin_zero[8];
};

struct in_addr {          /* Declared in in.h */
    unsigned long s_addr;
};
```


sa_family

OUTPUT. A structure member that identifies the address format. The corresponding field in the `sockaddr_in` structure is *sin_family*. The PRIMOS socket library supports only the `AF_INET` family.

sa_data

OUTPUT. A structure member that is a 14-byte character array containing the actual data value. The corresponding fields in the `sockaddr_in` structure are *sin_port*, *sin_addr*, and *sin_zero*.

namelen

INPUT/OUTPUT. Pointer to maximum size (in bytes) of the `sockaddr_in` structure. Initialize *namelen* to indicate the amount of space pointed to by *name* (ordinarily 16 bytes). When *namelen* is returned, it contains the actual size of the name returned (in bytes). If the call fails, the contents of *namelen* are not updated.

cc

OUTPUT. Return status. A value of 0 is returned if the call succeeds. If the call fails, a value of -1 is returned and the global variable *errno* (declared in `stdio.h`) provides more specific error information.

Errors

The following error codes describe why a `TUP$getsockname()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONTROLLERDOWN	The controller is not in operation.
ENOBUFS	The system did not have sufficient resources to perform the operation.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$getsockopt()

TUP\$getsockopt() retrieves the previously set values of options that are associated with a socket. Options include allowing a process to redefine socket address information by means of a TUP\$bind() call, allowing messages to be sent to a connected socket, and managing unsent messages if a TUP\$close_socket() is issued.

When the user program calls TUP\$getsockopt(), it must specify both the layer at which the option resides and the name of the option.

Use TUP\$setsockopt() to set options.

Call Syntax

```
#include <socket.h>
#include <tcp/ip_error.h>

cc = TUP$getsockopt(s, level, optname, optval, optlen);

long s, level, optname, cc;
char *optval;
long *optlen;
```

Parameters

s

INPUT. Socket descriptor returned by TUP\$socket().

level

INPUT. Layer on which the option is to be applied. Specify SOL_SOCKET for socket level options (this is the only level supported).

optname

INPUT. Option name. *optname* is a symbolic constant defined in the socket.h file. See the section entitled Socket Options below for a description of these symbolic constants.

optval

INPUT/OUTPUT. Pointer to the value of the option. The possible option values, described in Socket Options below, can be either short integers or structures. In calls to TUP\$getsockopt(), cast *optval* to a pointer to **char**.

optlen

INPUT/OUTPUT. Pointer to the size of the buffer for *optval*. *optlen* is modified on return to indicate the actual size of the *optval* returned.

cc

OUTPUT. Return status. A 0 is returned if the call succeeds. If the call fails, a value of -1 is returned and the global variable *errno* (declared in *stdio.h*) returns specific error information.

Socket Options

Socket level options, defined in the *socket.h* include file, control the operations of sockets. The following table of symbolic constants describes these options:

<i>Option</i>	<i>Description</i>
SO_REUSEADDR	Allows a process to redefine the socket address information using <i>TUP\$bind()</i> . A value of 1 means this option is set; a value of 0 means it is not set. (The default is 0.) Use this option in situations similar to the following: Process A has bound a socket to port 1010, completed a data transfer, and closed the socket. Process B immediately wants to bind a socket to port 1010. In order to do so, Process B must set the SO_REUSEADDR option before binding. If SO_REUSEADDR has been set, only a client SOCK_STREAM process can reuse a port, and it can do so only if and when a previous socket using that port has closed the connection or is still connected. A socket that is listening or that has not yet been connected cannot share a port with another socket.
SO_KEEPALIVE	Enables the periodic transmission of messages on a connected socket. If the connected process does not respond to these messages, the connection is considered broken and the processes using the socket are notified with an error returned. A value of 1 means this option is set; a value of 0 means it is not set. (The default is 0.)
SO_LINGER	Controls the actions taken when unsent messages are queued in the controller for a socket and a <i>TUP\$close_socket()</i> is performed. If the socket promises reliable delivery of data (that is, if it is of type SOCK_STREAM) and SO_LINGER is set, the system blocks the process on the close attempt until it can transmit data or until it decides that it cannot deliver the information. (Blocking occurs even if nonblocking I/O has been set by a <i>TUP\$ioctl()</i> call.) A timeout period, called the linger interval, is specified in the <i>TUP\$setsockopt()</i> call when SO_LINGER is requested. The linger interval is then passed to

the controller. If `SO_LINGER` is not set and a `TUP$close_socket()` is issued, the system processes the `TUP$close_socket()` so that the process can terminate as quickly as possible. The value for the `SO_LINGER` option is specified in the following structure:

```
struct linger {          /* Declared in socket.h */
    short l_onoff;        /* 1 for on, 0 for off */
    short l_linger;       /* Time in seconds */
};
```

SO_BUFSIZE

Indicates the number of bytes allocated to the socket by the protocol. This unsigned short integer has a minimum value of 0 (zero) and a maximum value of 65,536.

Errors

The following error codes describe why a `TUP$getsockopt()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONTROLLERDOWN	The controller is not in operation.
ENOBUFS	The system did not have sufficient resources to perform the operation.
ENOPROTOOPT	The option is unknown.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$ioctl()

TUP\$ioctl() sets and changes socket options that TUP\$setsockopt() does not support. TUP\$ioctl() enables the user program to change the characteristics of open socket descriptors.

Call Syntax

```
#include <ioctl.h>
#include <tcp/ip_error.h>

cc = TUP$ioctl(s, request, argp);

long s, request, cc;
char *argp;
```

Parameters

s

INPUT. Socket descriptor returned by TUP\$socket().

request

INPUT. An encoded request. The request is a macro definition in the ioctl.h file, either FIONREAD or FIONBIO. *request* specifies whether the list that the *argp* parameter specifies is to be read or written to for the socket specified by *s*.

FIONREAD	Gets the number of bytes or packets to read.
FIONBIO	Sets or clears nonblocking I/O.

argp

INPUT/OUTPUT. A pointer to the argument list needed for the requested operation. The argument must be a pointer to a long, which must then be cast to a pointer to char.

FIONREAD	The argument must be a pointer to a long, the contents of which are filled in with the number of bytes (for SOCK_STREAM) or packets (for SOCK_DGRAM) that are available.				
FIONBIO	The argument must be a pointer to a long. The contents of the pointer may be <table> <tr> <td>1</td> <td>Turns on nonblocking I/O.</td> </tr> <tr> <td>0</td> <td>Turns off nonblocking I/O. (Blocking I/O is the default.)</td> </tr> </table>	1	Turns on nonblocking I/O.	0	Turns off nonblocking I/O. (Blocking I/O is the default.)
1	Turns on nonblocking I/O.				
0	Turns off nonblocking I/O. (Blocking I/O is the default.)				

cc

OUTPUT. Contains the return status. A value of 0 is returned if the call succeeds. If an error occurs, a value of -1 is returned with the global variable *errno* (declared in *stdio.h*) set to indicate the error.

Errors

The following error codes describe why a `TUP$ioctl()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
EINVAL	The request is invalid.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$listen()

A server process that is ready to receive incoming connection requests from another process calls TUP\$listen(). Server processes that have called TUP\$listen() passively listen on a bound socket for connection requests from client processes.

TUP\$listen() specifies a backlog for incoming connections. The backlog specifies the maximum number of requests that can be queued. If the server process receives a connect request while the queue is full, the connection request is ignored but not refused. This response prompts the client process to retry the connection. No error message is returned.

TUP\$listen() applies to connection-based protocols (TCP). Therefore, TUP\$listen() applies only to sockets of type SOCK_STREAM. TUP\$listen() does not apply to SOCK_DGRAM (UDP) sockets.

Calling TUP\$listen() on an unbound socket is equivalent to calling TUP\$bind() with a name consisting of INADDR_ANY and a port of 0 (zero). Therefore, a server program on a 50 Series system that has only one LHC300 controller does not have to issue a TUP\$bind() call in order to listen on a socket. If the server omits the TUP\$bind() call, it may call TUP\$getsockname() in order to find out the socket name. The server must also make the socket name known to client programs that may be trying to reach it.

If the system has two controllers, and if the server wishes to receive connection requests on both controllers, the server program must create a socket for each controller, bind the proper Internet address to each socket, and then issue a TUP\$listen() call for each socket. Refer to the description of TUP\$bind().

Call Syntax

```
#include <tcp/ip_error.h>

cc = TUP$listen(s, backlog);

long s, backlog, cc;
```

Parameters

s

INPUT. Socket descriptor returned by TUP\$socket().

backlog

INPUT. Defines the maximum number of connections that can be queued simultaneously awaiting acceptance. The maximum size of the queue is 5.

cc

OUTPUT. Return status. If the call succeeds, 0 is returned. If the call fails, a value of -1 is returned and the global variable *errno* (declared in *stdio.h*) provides specific error information.

Errors

The following error codes describe why a `TUP$listen()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONTROLLERDOWN	The controller is not in operation.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
EOPNOTSUPP	The socket type (<code>SOCK_DGRAM</code>) does not support the <code>TUP\$listen()</code> operation.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$perror()

TUP\$perror() prints an error message at the terminal. This message describes the last error that was encountered during a call to an environment or socket subroutine.

TUP\$perror() uses the global variable *errno* (declared in *stdio.h*) to reference an array of character strings containing error messages. These character strings are located in the file *SYSOVL>TUP\$ERROR_TABLE*. For a list of the socket library error messages, see Appendix G.

To print error messages for non-socket routines, use the PRIMOS C library routine *perror()*. *redefine.h* does *not* provide a redefinition for TUP\$perror(); you must use the TUP\$ prefix when you call it.

Note

Users may also call the PRIMOS system subroutine *ER\$PRINT* to display error messages at the terminal. For information on using this subroutine, see Chapter 8. For an example of a call to *ER\$PRINT*, see Sample Program 2 in Chapter 10.

Call Syntax

```
#include <tcp/ip_error.h>
```

```
TUP$perror(s);
```

```
char *s;
```

Parameters

s

INPUT. A pointer to *char* containing the name of the program or routine that incurred the error.

TUP\$recv(), TUP\$recvfrom()

TUP\$recv() and TUP\$recvfrom() receive messages from a socket.

The TUP\$recv() call can receive data on a connected socket (type SOCK_STREAM) that has been connected by means of a call to TUP\$connect() or TUP\$accept(). It can also receive data on an unconnected socket (type SOCK_DGRAM) whose destination has been permanently specified by a call to TUP\$connect().

The TUP\$recvfrom() call can receive data only on an unconnected socket (type SOCK_DGRAM).

Call Syntax

```
#include <socket.h>
#include <tcp/ip_error.h>
#include <in.h>

cc = TUP$recv(s, buf, len, flags);

long s, cc;
char *buf;
long len, flags;

cc = TUP$recvfrom(s, buf, len, flags, from, fromlen);

long s, cc;
char *buf;
long len, flags;
struct sockaddr *from;
long *fromlen;

struct sockaddr {          /* Declared in socket.h */
    short sa_family;
    char sa_data[14];
};
```

Parameters

s
INPUT. Socket descriptor returned by TUP\$socket().

buf
INPUT/OUTPUT. Pointer to user message buffer area. When *buf* is returned, it contains the message received.

len

INPUT. Length (in bytes) of *buf*.

flags

INPUT. Set *flags* to 0 (zero) to indicate that normal data is to be received. Set *flags* to MSG_OOB to receive out-of-band data on SOCK_STREAM sockets. The program must also establish an on-unit (condition handler) to pick up the SIGURG signal. (See the sections entitled Transferring Data and Signals in Chapter 8.) If TUP\$recv() is called with MSG_OOB, previously sent lower-priority messages will be lost.

from

INPUT/OUTPUT. Pointer to sockaddr structure, the name of the socket from which the message was received. A program that calls TUP\$recvfrom() ordinarily declares an AF_INET socket name using the more specific sockaddr_in structure, and then casts its address to a pointer to a sockaddr structure. If the TUP\$recvfrom() call succeeds, the *sin_port* and *sin_addr* fields are filled in with the source address of the message. If the call fails, those fields are not updated. The sockaddr_in structure is declared as follows:

```
struct sockaddr_in {      /* Declared in in.h */
    short                sin_family;
    unsigned short       sin_port;
    struct in_addr       sin_addr;
    char                 sin_zero[8];
};

struct in_addr {          /* Declared in in.h */
    unsigned long        s_addr;
};
```

sa_family

INPUT/OUTPUT. A structure member that identifies the address format. The corresponding field in the sockaddr_in structure is *sin_family*. The PRIMOS socket library supports only the AF_INET family.

sa_data

INPUT/OUTPUT. A structure member that is a 14-byte character array containing the actual data value. The corresponding fields in the sockaddr_in structure are *sin_port*, *sin_addr*, and *sin_zero*.

fromlen

INPUT/OUTPUT. Pointer to maximum size (in bytes) of *from*. Initialize the contents of *fromlen* to indicate the amount of space pointed to by *from* (ordinarily 16 bytes). When *from* is returned, *fromlen* points to the actual size of the returned address (in bytes). If the call fails, the contents of *fromlen* are not updated.

cc

OUTPUT. Return status. The length of the message is returned in *cc*. If a message is too long to fit in the supplied buffer, and if the socket type is `SOCK_DGRAM`, excess bytes may be discarded. If *cc* is 0, it means that the remote host has closed the connection. If no messages have arrived at the socket, the `TUP$recv()` or `TUP$recvfrom()` call waits for a message to arrive, unless the socket is nonblocking. If the socket is nonblocking, the value of *cc* is -1 and the external variable *errno* (declared in `stdio.h`) is set to `EWouldBlock`.

Errors

The following error codes describe why a `TUP$recv()` or `TUP$recvfrom()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONTROLLERDOWN	The controller is not in operation.
EINTR	The receive was interrupted by delivery of a signal before any data was available for the receive.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.
EWouldBlock	The socket is marked nonblocking and the receive operation would block.

TUP\$select()

TUP\$select() examines the state of multiple sockets simultaneously. The call enables a process to wait on multiple sockets for receiving data, sending data, and signals (exceptions).

Signals include out-of-band data, a controller failure, or a remote machine closing a socket. A TUP\$select() with the exception fields set may be used to identify on which socket the signal occurred. For information about signals, see Chapter 8.

TUP\$select() takes as arguments three bit masks. Another argument (*nfds*) specifies the number of significant bits in the mask fields. The maximum length of the mask is 32 bits.

Each bit position in the mask represents a socket. For example, bit 32 (least significant bit) represents socket 0, bit 31 represents socket 1, and so on. TUP\$select() examines the descriptors that the user program specifies. In general, a socket descriptor is selected if a 1 is present in the bit of the mask representing that socket.

If an operation is possible on the specified socket, TUP\$select() updates the bit masks and returns the number of such descriptors.

The user program can specify a timeout value if the selection is not to last more than a predetermined period of time.

Use TUP\$select() to check for pending requests before each TUP\$accept().

Call Syntax

```
#include <tcp/ip_time.h>
#include <tcp/ip_error.h>

cc = TUP$select(nfds, readfds, writefds, exceptfds, timeout);

long  nfds, cc;
char  *readfds, *writefds, *exceptfds;
struct timeval *timeout;

struct timeval {           /* Declared in tcp/ip_time.h */
    long  tv_sec;
    long  tv_usec;
};
```

Parameters

nfds

INPUT. Specifies the number of significant bits in the fields of the *readfds*, *writefds*, and *exceptfds* parameters, described below. The limit is 32 bits.

readfds

INPUT/OUTPUT. Pointer to the bit mask of sockets from which the caller wants to receive data. On return, the bits of *readfds* are set to indicate which sockets are ready for receiving. If *readfds* includes a socket that is used in a TUP\$listen() call, then the returned value in *readfds* indicates whether any connections are pending.

writefds

INPUT/OUTPUT. Pointer to the bit mask of sockets on which the caller wants to send data. On return, the bits of *writefds* are set to indicate which sockets are ready for sending.

exceptfds

INPUT/OUTPUT. Pointer to the bit mask of sockets for which exceptional conditions (such as out-of-band data) are pending. On return, the bits of *exceptfds* are set to indicate sockets with exceptions.

timeout

INPUT. Pointer to timeout period structure. The *tv_sec* and *tv_usec* fields refer to seconds and microseconds, respectively. If both fields are 0, the selection takes the form of a poll that returns immediately. TUP\$select() polls all the sockets and returns the appropriate values in the bit mask parameters. If *timeout* is set to NULL, the selection is blocked indefinitely. That is, *cc* is returned only when a socket can be selected.

cc

OUTPUT. Return status. TUP\$select() returns one of the following values:

- Number of sockets that are contained in the bit masks.
- -1, with the global variable *errno* (declared in *stdio.h*) set to one of the errors described in the next section.
- 0 (zero) if the call returned due to the timeout expiring.

Errors

The following error codes describe why a TUP\$select() call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	One of the masks contains a descriptor not specified by <i>nfds</i> .
EINTR	The TUP\$select() was interrupted by an unrequested event.
EINVAL	The specified time limit is unacceptable.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$send(), TUP\$sendto()

TUP\$send() and TUP\$sendto() send messages from a socket.

The TUP\$send() call can send data to a connected socket (type SOCK_STREAM) that has been connected by means of a call to TUP\$connect() or TUP\$accept(). It can also send data to a SOCK_DGRAM socket, if a TUP\$connect() call has specified the destination of the data.

The TUP\$sendto() call can send data only to an unconnected socket (type SOCK_DGRAM). If TUP\$sendto() is called on an unbound socket, TUP\$sendto() binds the socket.

If the socket does not have message space to hold the message to be transmitted, TUP\$send() blocks unless the socket has been placed in nonblocking I/O mode. (Use the TUP\$ioctl() routine to mark a socket nonblocking.)

TUP\$send() does not indicate when it cannot deliver data.

Call Syntax

```
#include <socket.h>
#include <tcp/ip_error.h>
#include <in.h>
```

```
cc = TUP$send(s, msg, len, flags);
```

```
long s, cc;
char *msg;
long len, flags;
```

```
cc = TUP$sendto(s, msg, len, flags, to, tolen);
```

```
long s, cc;
char *msg;
long len, flags;
struct sockaddr *to;
long *tolen;
```

```
struct sockaddr {      /* Declared in socket.h */
    short sa_family;
    char sa_data[14];
};
```

Parameters

s
INPUT. Socket descriptor returned by TUP\$socket().

msg

INPUT. Pointer to the message to be transmitted.

len

INPUT. Length (in bytes) of the message. In the case of SOCK_DGRAM, if the message is too long to pass through the underlying protocol, the error EMSGSIZE is returned, and the message is not transmitted. In the case of SOCK_STREAM, the message is broken up by the underlying protocols and delivered in sequence.

flags

INPUT. Set *flags* to 0 (zero) to send normal data. Set *flags* to MSG_OOB to send out-of-band data on SOCK_STREAM sockets.

to

INPUT. Pointer to name of socket to which message is to be sent. A program that calls TUP\$sendto() ordinarily declares an AF_INET socket name using the more specific sockaddr_in structure, and then casts its address to a pointer to a sockaddr structure. The sockaddr_in structure is declared as follows:

```
struct sockaddr_in {    /* Declared in in.h */
    short              sin_family;
    unsigned short     sin_port;
    struct in_addr      sin_addr;
    char               sin_zero[8];
};

struct in_addr {        /* Declared in in.h */
    unsigned long       s_addr;
};
```

sa_family

INPUT. A structure member that identifies the address format. The corresponding field in the sockaddr_in structure is *sin_family*. The PRIMOS socket library supports only the AF_INET family.

sa_data

INPUT. A structure member that is a 14-byte character array containing the actual data value. The corresponding fields in the sockaddr_in structure are *sin_port*, *sin_addr*, and *sin_zero*.

to len

INPUT. Pointer to size (in bytes) of *to* (ordinarily 16 bytes).

cc

OUTPUT. Return status. The call returns the number of characters sent or returns `-1` if an error is detected locally. The specific error code is placed in the global variable *errno* (declared in *stdio.h*).

Errors

The following error codes describe why a `TUP$send()` or `TUP$sendto()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONNREFUSED	A call to <code>TUP\$sendto()</code> has failed. The host may be down or nonexistent, or the network may be unreachable.
ECONTROLLERDOWN	The controller is not in operation.
EINVAL	One of the arguments is invalid.
EMSGSIZE	The socket type (<code>SOCK_DGRAM</code>) requires that the message be sent as a single entity. The size of the message violates this requirement.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.
EUBIERR	The connection to the controller has not been established or has been interrupted.
EWouldBLOCK	The socket is marked nonblocking and the send operation would block.

TUP\$setsockopt()

TUP\$setsockopt() sets options that are associated with a socket. Options include allowing a process to redefine socket address information by means of a TUP\$bind() call, allowing messages to be sent to a connected socket, and managing unsent messages if a TUP\$close_socket() is issued.

While options can exist at more than one protocol level, they are always present at the uppermost, socket level. When the user program calls TUP\$setsockopt(), it must specify both the level at which the option resides and the name of the option.

Use TUP\$getsockopt() to retrieve the values of options that have been set with TUP\$setsockopt().

The socket.h include file contains definitions for socket options.

Call Syntax

```
#include <socket.h>
#include <tcp/ip_error.h>

cc = TUP$setsockopt(s, level, optname, optval, optlen);

long s, level, optname, cc;
char *optval;
long optlen;
```

Parameters

s

INPUT. Socket descriptor returned by TUP\$socket().

level

INPUT. Level on which the option is to be applied. Specify SOL_SOCKET for socket level options (this is the only level supported).

optname

INPUT. Option name. *optname* is a symbolic constant defined in the socket.h file and described in the section entitled Socket Options below. *optname* and its associated value are passed uninterpreted to the appropriate protocol module to be interpreted.

optval

INPUT. Pointer to the value of the option. The possible option values, described in Socket Options below, can be either short integers or structures. In calls to `TUP$setsockopt()`, *optval* must be cast to a pointer to `char`. The value of the option usually indicates whether the option is to be turned on or off. If no option value is to be set, set *optval* to `(char *)NULL`.

optlen

INPUT. Size of *optval*.

cc

OUTPUT. Return status. A value of 0 is returned if the call succeeds. If the call fails, a value of -1 is returned and the global variable *errno* (declared in `stdio.h`) returns specific error information.

Socket Options

Socket level options, defined in the `socket.h` include file, control the operations of sockets. The following table of symbolic constants describes these options:

<i>Option</i>	<i>Description</i>
SO_REUSEADDR	Allows a process to redefine the socket address information using <code>TUP\$bind()</code> . Specify 1 to turn <code>SO_REUSEADDR</code> on, 0 to turn it off. The default is off. Use this option in situations similar to the following: Process A has bound a socket to port 1010, completed a data transfer, and closed the socket. Process B immediately wants to bind a socket to port 1010. In order to do so, Process B must set the <code>SO_REUSEADDR</code> option before binding. If <code>SO_REUSEADDR</code> has been set, only a client <code>SOCK_STREAM</code> process can reuse a port, and it can do so only if and when a previous socket using that port has closed the connection or is still connected. A socket that is listening or that has not yet been connected cannot share a port with another socket.
SO_KEEPALIVE	Enables the periodic transmission of messages on a connected socket. If the connected process does not respond to these messages, the connection is considered broken and the processes using the socket are notified with an error returned. Specify 1 to turn <code>SO_KEEPALIVE</code> on, 0 to turn it off. The default is off.
SO_LINGER	Controls the actions taken when unsent messages are queued in the controller for a socket and a <code>TUP\$close_socket()</code> is performed. If the socket promises reliable delivery of data and <code>SO_LINGER</code> is set, the system blocks the process on the close

attempt until it can transmit data or until it decides that it cannot deliver the information. (Blocking occurs even if nonblocking I/O has been set by a TUP\$ioctl() call.) A timeout period, called the linger interval, is specified in the TUP\$setsockopt() call when SO_LINGER is requested. The linger interval is then passed to the controller. The value for the SO_LINGER option is specified in the following structure:

```
struct linger {          /* Declared in socket.h */
    short l_onoff;        /* 1 for on, 0 for off */
    short l_linger;       /* Time in seconds */
};
```

By default, both fields of this structure are 0.

SO_BUFSIZE

Indicates the number of bytes allocated to the socket by the protocol. This unsigned short integer has a minimum value of 0 (zero) and a maximum of 65,536.

Errors

The following error codes describe why a TUP\$setsockopt() call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONTROLLERDOWN	The controller is not in operation.
ENOPROTOOPT	The option is unknown.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$shutdown()

TUP\$shutdown() causes all or part of a UDP (SOCK_DGRAM) connection on the socket to be shut down. For a SOCK_DGRAM socket, TUP\$shutdown() affects activity only on the local host. No information is forwarded to the remote host.

For a TCP (SOCK_STREAM) socket, TUP\$shutdown() performs part of the action of TUP\$close_socket(); it closes the remote connection. For this reason, it is usually unnecessary to use TUP\$shutdown() on a SOCK_STREAM socket. If a user program wants to close a SOCK_STREAM socket, it should call TUP\$close_socket(). Applying TUP\$shutdown() to a SOCK_STREAM socket may cause any data queued for that socket to be discarded immediately.

Call Syntax

```
#include <tcp/ip_error.h>

cc = TUP$shutdown(s, how);

long s, how, cc;
```

Parameters

s

INPUT. Socket descriptor returned by TUP\$socket().

how

INPUT. Type of shutdown. This parameter is valid only for a SOCK_DGRAM socket. *how* can have the following values:

- 0 The user program no longer wants to receive data.
- 1 The user program no longer wants to send data.
- 2 The user program no longer wants to send or receive data.

cc

OUTPUT. Return status. If the call succeeds, 0 is returned. If the call fails, a value of -1 is returned and the global variable *errno* (declared in *stdio.h*) provides specific error information.

Errors

The following error codes describe why a `TUP$shutdown()` call can fail:

<i>Error Code</i>	<i>Description</i>
EBADF	<i>s</i> is not a valid socket descriptor.
ECONTROLLERDOWN	The controller is not in operation.
EINVAL	The <i>how</i> parameter is invalid.
ENOTCONN	The specified socket is not connected.
ENOTSOCK	The descriptor <i>s</i> is not an active socket.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

TUP\$socket()

TUP\$socket() creates a socket in a specified domain and associates the socket with a protocol. The user program is returned a socket descriptor that it uses to reference the socket in subsequent socket calls.

Socket level options, defined in the socket.h include file, control the operations of sockets. See TUP\$getsockopt() and TUP\$setsockopt() for a description of these options.

Call Syntax

```
#include <socket.h>
#include <tcp/ip_error.h>

s = TUP$socket(af, type, protocol);

long  af, type, protocol, s;
```

Parameters

af

INPUT. Specifies a domain address format in which the socket is to be created. Addresses that are specified in later operations and that use the socket must be interpreted with this format. These formats are defined in the socket.h include file. Only the DARPA Internet format (specified by the constant AF_INET) is supported.

type

INPUT. Socket type specifies the semantics of communications. Currently defined types are

- SOCK_STREAM — stream type socket
- SOCK_DGRAM — datagram type socket

protocol

INPUT. Protocol to be used with the socket. Enter a value of zero. TUP\$socket() specifies a valid protocol from protocols that comprise the communication domain and that support the requested socket type.

The communication domain determines the protocol number that identifies the protocol. Normally, only a single protocol exists to support a particular socket type that uses a given address format. The TCP protocol supports SOCK_STREAM. The UDP protocol supports SOCK_DGRAM.

s

OUTPUT. The returned socket descriptor, a nonnegative integer. If the call fails, *s* is set to `-1` and the global variable *errno* (declared in `stdio.h`) provides the specific error code.

Errors

The following error codes describe why a `TUP$socket()` call can fail:

<i>Error Code</i>	<i>Description</i>
EAFNOSUPPORT	The specified address family is not supported in this version of the socket library.
ENOBUFS	The system did not have sufficient resources to perform the operation.
EPROTONOSUPPORT	The specified protocol is not supported.
ESOCKTNOSUPPORT	The specified socket type is not supported in this address family.
ESYSERR	A system error has occurred. Contact your System or Network Administrator.

Network Library Subroutine Calls

This section describes subroutine calls that

- Map host names to Internet addresses
- Map Internet addresses to network numbers
- Convert Internet addresses from character strings to integers and back again
- Obtain controller device addresses

Table 9-4 describes how you can use these subroutine calls to retrieve the information your application may require.

Table 9-4
How to Use the Network Library Subroutines

<i>If you have</i>	<i>And you want to know</i>	<i>Use</i>
No information	The local host name	TUP\$gethostname()
A host name	One of the host's Internet addresses in unsigned long form	TUP\$gethostbyname()
A host name	All of the host's Internet addresses in unsigned long form	TUP\$getalladdrs()
Internet address of host in unsigned long form	Host device addresses and Internet addresses	TUP\$getdevaddrs()
Host device addresses and Internet addresses	Broadcast address of controller	TUP\$getbroadcastaddr()
Internet address in unsigned long form	Host name	TUP\$gethostbyaddr()
Internet address in character string form	Internet address in unsigned long form	TUP\$inet_addr()
Internet address in structure form	Local address in unsigned long form	TUP\$inet_lnaof()
Internet address in structure form	Network number in unsigned long form	TUP\$inet_netof()
Internet address in structure form	Internet address in character string form	TUP\$inet_ntoa()
Network number and local address in unsigned long form	Internet address in structure form	TUP\$inet_makeaddr()
Network number in character string form	Network number in unsigned long form	TUP\$inet_network()

TUP\$getalladdrs()

TUP\$getalladdrs() returns all of the Internet addresses of a requested host if the call is given any of the alternate names of a host. It uses the client portion of the Hostname Service protocol (described in Chapter 12) to perform the search. First, the local HOSTS.TXT configuration file is examined. If this file does not contain the information, a remote server is contacted if one is available.

The call returns a pointer to a `mult_addrs` structure. The returned `mult_addrs` structure is defined as static within TUP\$getalladdrs(). Therefore, you must copy the data to another area if you intend to use it after a second call to TUP\$getalladdrs().

Call Syntax

```
#include <inet.h>
#include <netdb.h>

cc = TUP$getalladdrs(name);

char *name;
struct mult_addrs *cc;

struct mult_addrs {      /* Declared in netdb.h */
    short num_addrs;
    unsigned long addr_array[MAXADDRES];
};
```

Parameters

name

INPUT. A pointer to **char** that must point to the official name or any of the aliases of the host about which you want to obtain information.

cc

OUTPUT. A pointer to a `mult_addrs` structure. If the call fails, it returns a NULL pointer.

num_addrs

OUTPUT. A structure member that contains the number of valid addresses in *addr_array*. The value of *num_addrs* is always less than or equal to MAXADDRES, a constant. MAXADDRES is defined as 4. If a host has more than four addresses, TUP\$getalladdrs() returns only the first four addresses that it finds. This should not be a problem, because it is very unlikely that a host would have more than two or three addresses.

addr_array

OUTPUT. A structure member that contains the valid Internet addresses of the requested hosts.

TUP\$getbroadcastaddr()

TUP\$getbroadcastaddr() returns the **broadcast address** for a controller. A message sent to a broadcast address is directed to all hosts on the network rather than to a single specific host.

An Internet address is reserved for broadcast if its local address part consists of all 1s. For example, on the Class A network 10 (the ARPANET proper), the broadcast address is 10.255.255.255.

Call Syntax

```
#include <netdb.h>
#include <inet.h>

broadaddr = TUP$getbroadcastaddr(device_ptr);

unsigned long broadaddr;
struct dev_addr_entry *device_ptr;

struct dev_addr_entry {      /* Declared in netdb.h */
    short mflag;
    short dev_addr;
    unsigned long internet_address;
    unsigned long subnet_mask;
};
```

Parameters

device_ptr

INPUT. A pointer to a dev_addr_entry structure. A call to TUP\$getdevaddrs() returns a pointer to an array of these structures; use the address of one of these as the argument to TUP\$getbroadcastaddr().

mflag

INPUT. A structure member that is set to 1 or 0 by TUP\$getdevaddrs().

dev_addr

INPUT. A structure member that indicates the device address of the LHC300 controller.

internet_address

INPUT. A structure member that indicates the Internet address of the LHC300 controller.

subnet_mask

INPUT. A structure member that indicates the subnetwork mask of the LHC300 controller, if it has one. If the controller has no subnetwork mask, this field is set to FF000000 for a Class A network, FFFF0000 for a Class B network, or FFFFFFF0 for a Class C network.

broadaddr

OUTPUT. The returned value, an unsigned long integer that contains the broadcast address for the controller. The broadcast address is formed by ORing the Internet address with the one's complement of the subnetwork mask.

TUP\$getdevaddrs()

TUP\$getdevaddrs() supplies information that user programs need in order to call TUP\$setup_socket_env(). In particular, it returns the device addresses of a system's LHC300 controllers.

TUP\$getdevaddrs() also supports client programs that make TCP/IP connect requests on systems that have two LHC300 controllers. The user program must know which controller to use to initiate the call. TUP\$getdevaddrs() returns LHC300 device addresses that can be used to reach a particular destination.

TUP\$getdevaddrs() is passed the Internet address of a host. For server programs, the argument is the address of the local host; for client programs, it is the address of the destination host. The routine then checks the Internet addresses of the LHC300 controllers. It compares the network number part of the destination host address with the network number part of the controller addresses. The routine places the controller device addresses in an array, in an order determined by the results of the comparison.

Depending upon the results of the comparison, the routine does the following:

- If one controller address is matched, but not the other, the device address of the controller with the matching network number is placed in the first entry of the array. The other device address is placed second in the array. Try the second controller if the first controller is not up or is not accepting call requests. If the second controller is not on the same network, a gateway or set of gateways must provide a path to the destination.
- If both controller addresses match, the device addresses of the two controllers are placed in the first two positions in the array.
- If neither controller address matches, the routine places the device addresses in the array, but in no particular order. The routine does not attempt to analyze gateway connections to find a path to the destination.

The calling program should always attempt the call with the first device address in the array. If this fails, it should substitute the second. If the calling program tries both addresses unsuccessfully, the calling program must give up.

TUP\$getdevaddrs() returns a pointer to a structure shown below. The structure is declared static. You must copy data out of it before making a second call to the routine.

Call Syntax

```
#include <inet.h>
#include <netdb.h>
```

```
cc = TUP$getdevaddrs(addr);
```

```
unsigned long  addr;
struct dev_addr *cc;

struct dev_addr {          /* Declared in netdb.h */
    short num_devs;
    struct dev_addr_entry dev_addr_array[MAXLHC];
};

struct dev_addr_entry {    /* Declared in netdb.h */
    short mflag;
    short dev_addr;
    unsigned long internet_address;
    unsigned long subnet_mask;
};
```

Parameters

addr

INPUT. An unsigned long integer that holds an Internet address. For client programs, *addr* is the address of the destination host; for server programs, *addr* is the address of the local host.

cc

OUTPUT. Returned value. If TUP\$getdevaddrs() succeeds, it returns a pointer to a dev_addr structure. If the call fails, it returns a NULL pointer.

num_devs

OUTPUT. A structure member that indicates the number of controllers on the system.

dev_addr_array

OUTPUT. A structure member that is an array of device addresses. *dev_addr_array* always contains a number of device addresses equal to the number of controllers on the system (either 1 or 2). The addresses are sorted in the array in the order in which they should be used in placing the call. That is, use the first entry first. If that controller is shut down or is not accepting call requests for some reason, try the second. MAXLHC is defined as 8. PRIMOS TCP/IP supports only two controllers.

mflag

OUTPUT. A structure member that is set to 1 if the LHC300 controller's network number matches that of *addr*, and is set to 0 (zero) if the controller's network number does not match that of *addr*.

dev_addr

OUTPUT. A structure member that indicates the device address of the LHC300 controller.

internet_address

OUTPUT. A structure member that indicates the Internet address of the LHC300 controller.

subnet_mask

OUTPUT. A structure member that indicates the subnetwork mask of the LHC300 controller, if it has one. If the controller has no subnetwork mask, this field is set to FF000000 for a Class A network, FFFF0000 for a Class B network, or FFFFFFF0 for a Class C network.

TUP\$gethostbyaddr()

TUP\$gethostbyaddr() returns the name or names of a host when it is given any of the host's Internet addresses. It uses the client portion of the Hostname Service protocol (described in Chapter 12) to perform the search. First, the local HOSTS.TXT configuration file is examined. If this file does not contain the information, a remote server is contacted if one is available.

The user program cannot supply this routine one Internet address of a host in order to find all the Internet addresses of that host. Instead, use TUP\$gethostbyaddr() to find the host name and then call TUP\$getalladdrs() to find all the Internet addresses.

Note

TUP\$gethostbyaddr() is modeled on a 4.3BSD routine. It retains the same calling sequence and returns a pointer to the same type of structure as does the Berkeley version. The call and the structure returned are complicated because Berkeley supports both DARPA Internet and Xerox NS protocols.

TUP\$gethostbyaddr() returns a pointer to a structure described below. The returned structure is defined as static within TUP\$gethostbyaddr(). Therefore, you must copy the data to another area if you intend to use it after a second call to TUP\$getalladdrs().

Call Syntax

```
#include <in.h>
#include <inet.h>
#include <netdb.h>
#include <socket.h>

cc = TUP$gethostbyaddr(addr, len, type);

char *addr;
long len, type;
struct hostent *cc;

struct hostent {          /* Declared in netdb.h */
    char *h_name;
    char **h_aliases;
    long h_addrtype;
    long h_length;
    char **h_addr_list;
#define h_addr h_addr_list[0]
};
```


Parameters

addr

INPUT. A pointer to **char**. The argument should be a pointer to the unsigned long form of the Internet address cast as a pointer to **char**. For example,

```
unsigned long    tempaddr;
```

```
TUP$gethostbyaddr((char *) &tempaddr, AF_INET_ADDR_LEN, AF_INET);
```

len

INPUT. The length of the Internet address in bytes. *len* is always `AF_INET_ADDR_LEN`, because Internet addresses are always this length. (`AF_INET_ADDR_LEN` is defined as 4 in the `in.h` file).

type

INPUT. Address type. Set *type* to `AF_INET` to specify that Internet type addresses are being used. (`AF_INET` is defined in `socket.h`.)

cc

OUTPUT. Returned value. A pointer to a `hostent` structure. If the call fails, it returns a `NULL` pointer.

h_name

OUTPUT. In the returned structure, *h_name* is a pointer to **char** containing the official name of the host.

h_aliases

OUTPUT. In the returned structure, *h_aliases* is an array of pointers to **char**. There is a maximum of 4 aliases. The first element of the array contains the first alias in the alias list; the second element contains the second alias, and so on. A `NULL` pointer in the array marks the end of the array and indicates that there are no more aliases.

h_addrtype

OUTPUT. In the returned structure, *h_addrtype* is `AF_INET`, the type of address.

h_length

OUTPUT. In the returned structure, *h_length* is the length of the address in bytes.

h_addr_list

OUTPUT. In the returned structure, *h_addr_list* is a list of Internet addresses from the remote host configuration file. The definition of *h_addr_list* makes it appear that it is an array of pointers to **char**, each of which contains a different Internet address. This is not the case. Only the first element of the array, *h_addr*, described below, is actually returned. The Internet address pointed to is always the same as the address specified in the *addr* argument.

h_addr

OUTPUT. In the returned structure, a pointer to **char** that actually contains the Internet address expressed as an unsigned long integer. The contents, if needed, can be retrieved in the following way:

```
unsigned long  retaddr;

retaddr = * (unsigned long *) cc->h_addr;
```

TUP\$gethostbyname()

TUP\$gethostbyname() returns an Internet address (hostent structure) when given the name of a remote host. (If you want all of a host's Internet addresses, call TUP\$getalladdrs().) TUP\$gethostbyname() is similar to TUP\$gethostbyaddr(), but it searches the HOSTS.TXT file (the configuration file that contains the list of remote hosts) for the host name, either an official name or any of the aliases. If a host has more than one Internet address, the hostent structure that the routine returns describes just one of them. TUP\$gethostbyname() searches the HOSTS.TXT file from the beginning of the file until it either finds a matching host name or reaches the end of the file. If no match is found in HOSTS.TXT, TUP\$gethostbyname() contacts a remote server to perform the search.

Note

TUP\$gethostbyname() is modeled on a 4.3BSD routine. It retains the same calling sequence and returns a pointer to the same type of structure as does the Berkeley version. The call and the structure returned are complicated because Berkeley supports both DARPA Internet and Xerox NS protocols.

TUP\$gethostbyname() returns a pointer to the structure described below. The returned structure is defined as static within TUP\$gethostbyname(). Therefore, you must copy the data to another area if you intend to use it after a second call to TUP\$gethostbyname().

Call Syntax

```
#include <inet.h>
#include <netdb.h>

cc = TUP$gethostbyname(name);

char *name;
struct hostent *cc;

struct hostent {          /* Declared in netdb.h */
    char *h_name;
    char **h_aliases;
    long h_addrtype;
    long h_length;
    char **h_addr_list;
#define h_addr h_addr_list[0]
};
```

Parameters

name

INPUT. The official name or alias name of the host to be searched.

cc

OUTPUT. Returned value. A pointer to a `hostent` structure. If the call fails, it returns a `NULL` pointer.

h_name

OUTPUT. In the returned structure, *h_name* is a pointer to `char` that contains the official name of the host.

h_aliases

OUTPUT. In the returned structure, *h_aliases* is an array of pointers to `char`. There is a maximum of 4 aliases. The first element in the array points to the first alias in the alias list; the second element points to the second alias, and so on. A `NULL` pointer in the array marks the end of the array and indicates that there are no more aliases.

h_addrtype

OUTPUT. In the returned structure, *h_addrtype* is `AF_INET`, the type of address.

h_length

OUTPUT. In the returned structure, *h_length* is the length of the address in bytes.

h_addr_list

OUTPUT. In the returned structure *h_addr_list* is a list of Internet addresses from the remote host configuration file. The definition of *h_addr_list* makes it appear that it is an array of pointers to `char`, each of which contains a different Internet address. This is not the case. Only the first element of the array, *h_addr*, described below, is actually returned.

h_addr

OUTPUT. In the returned structure, a pointer to `char` that actually contains the Internet address expressed as an unsigned long integer. The contents can be retrieved in the following way:

```
unsigned long  retaddr;

retaddr = * (unsigned long *) cc->h_addr;
```

TUP\$gethostname()

TUP\$gethostname() returns the name of the local 50 Series host entered in the THISHOST file.

Call Syntax

```
#include <inet.h>
```

```
cc = TUP$gethostname(name, namelen);
```

```
char *name;  
long namelen, cc;
```

Parameters

name

INPUT/OUTPUT. A pointer to `char`. On return, TUP\$gethostname() returns the name of the local host (up to *namelen* characters).

namelen

INPUT. The size of the *name* array.

cc

OUTPUT. Returned value. If TUP\$gethostname() succeeds, it returns a value of 0 (zero). If the call fails, it returns a value of -1.

TUP\$inet_addr()

TUP\$inet_addr() interprets a character string in dot notation and returns an unsigned long integer. (Internet notation is described in Chapter 12.)

Call Syntax

```
#include <inet.h>
```

```
cc = TUP$inet_addr(cp);
```

```
char *cp;  
unsigned long cc;
```

Parameters

cp

INPUT. Pointer to a character representation of an Internet address in dot notation.

cc

OUTPUT. Returned value. TUP\$inet_addr() returns the Internet address reduced to a 32-bit unsigned long integer.

TUP\$inet_lnaof()

TUP\$inet_lnaof() processes Internet addresses and returns a local network address.

TUP\$inet_lnaof() takes an Internet address as an `in_addr` structure and strips away the network number. If any subnet bits are set, it also strips these away. It returns the local address as an unsigned long integer. It works on Class A, B, and C Internet addresses. (Internet address classes are described in Chapter 12.)

Call Syntax

```
#include <in.h>
#include <inet.h>

cc = TUP$inet_lnaof(in);

struct in_addr in;
unsigned long cc;

struct in_addr {          /* Declared in in.h */
    unsigned long s_addr;
};
```

Parameters

in

INPUT. A structure containing one member, which is a Class A, B, or C Internet address.

cc

OUTPUT. Returned value. TUP\$inet_lnaof() returns the local host address as an unsigned long integer.

TUP\$inet_makeaddr()

TUP\$inet_makeaddr() creates an Internet address from two integers representing the network number and the local host address.

Call Syntax

```
#include <in.h>
#include <inet.h>

cc = TUP$inet_makeaddr(net, host);

long net, host;
struct in_addr cc;

struct in_addr {          /* Declared in in.h */
    unsigned long s_addr;
};
```

Parameters

net

INPUT. Network number.

host

INPUT. Local host number.

cc

OUTPUT. Returned value. TUP\$inet_makeaddr() returns an Internet address expressed as an in_addr structure.

TUP\$inet_netof()

TUP\$inet_netof() takes an Internet address as an `in_addr` structure and returns just the network number part as an unsigned long integer.

Call Syntax

```
#include <in.h>
#include <inet.h>

cc = TUP$inet_netof(in);

struct in_addr in;
unsigned long cc;

struct in_addr {          /* Declared in in.h */
    unsigned long s_addr;
};
```

Parameters

in

INPUT. A structure containing one member, which is an Internet address.

cc

OUTPUT. Returned value. TUP\$inet_netof() returns the network number as an unsigned long integer.

TUP\$inet_network()

TUP\$inet_network() takes a character representation of a network number and reduces it to an unsigned long integer.

Call Syntax

```
#include <inet.h>
```

```
cc = TUP$inet_network(cp);
```

```
char *cp;  
unsigned long cc;
```

Parameters

cp

INPUT. A pointer to a character representation of a network number in dot notation. The number can refer to a Class A, B, or C network.

cc

OUTPUT. Returned value. TUP\$inet_network() returns the address reduced to a 32-bit unsigned integer.

TUP\$inet_ntoa()

TUP\$inet_ntoa() takes an Internet address specified as an `in_addr` structure and returns it as a character array in dot notation. The array is defined as static within TUP\$inet_ntoa(). Therefore, you must copy the character array to another area if you intend to use it after a second call to TUP\$inet_ntoa().

Call Syntax

```
#include <in.h>
#include <inet.h>

cc = TUP$inet_ntoa(in);

struct in_addr in;
char *cc;

struct in_addr {          /* Declared in in.h */
    unsigned long s_addr;
};
```

Parameters

in

INPUT. An Internet address specified as a structure.

cc

OUTPUT. Returned value. TUP\$inet_ntoa() returns a pointer to `char`. Declare the buffer to hold the returned value as a character array of length 16.

Sample Programs

This chapter provides two fully commented sample network applications written in the C programming language. The programs demonstrate how an application that calls the PRIMOS socket library can communicate with another application on a remote host. Both programs use SOCK_STREAM (TCP) sockets.

The sample client program, TCP_CALENDAR_CLIENT.C, sends a message to the server program requesting the time of day. The sample server program, TCP_CALENDAR_SERVER.C, listens for requests and sends back the time to the client.

Sample Program 1: Client Program

The client program performs the following steps:

1. Obtains information about the local host.
2. Sets up the socket environment.
3. Creates a socket and binds the local host address to it.
4. Establishes a connection to the server.
5. Sends a request message and receives a reply.
6. Closes the socket.
7. Closes the socket environment.

Commented Program

```
/*
 * TCP_CALENDAR_CLIENT.C:   example of a TCP client process.
 */

/*
 * Include files.
 */

#include <stdio.h>
/* PRIMOS C I/O header file. */
```

```
#include <time.h>
    /* Header file for time() and localtime(). */

#include <socket.h>
    /* Socket-related definitions. */

#include <ioctl.h>
    /* Definitions for ioctl function. */

#include <in.h>
    /* Internet constants and structures. */

#include <inet.h>
    /* External declarations for network library routines. */

#include <netdb.h>
    /* Structures used by network library routines. */

#include <tcp/ip_time.h>
    /* Time structures from BSD. */

#include <tcp/ip_error.h>
    /* Error code definitions for socket routines. */

#include <redefine.h>
    /* Allows program to omit TUP$ prefix on routine names. */

/* Socket declarations */
extern long    bind();
extern long    close_socket();
extern long    close_socket_env();
extern long    connect();
extern long    recv();
extern long    send();
extern long    setup_socket_env();
extern long    socket();

static char    req[] = "What time is it?";

/*
 * Start of code
 */
calendar_client()
{
    long i;                /* General purpose index. */
    long cc;               /* Socket condition code. */
    long s;                /* Socket identifier. */

    unsigned long inet_address; /* Internet address. */
    char rx_buff[80];        /* Buffer to receive TOD. */
    char hostname[6];        /* Host name (PRIME). */
    struct sockaddr_in my_name; /* Socket name. */
    long namelen = sizeof(my_name); /* Name length. */
    struct sockaddr_in rem_name; /* Remote socket name. */
    struct mult_addr *mult_addr_ptr; /* Internet address structure. */
    struct dev_addr *dev_addr_ptr; /* Device address structure. */
}
```

```

CONTROLLER_LIST_ENTRY controller_list[MAXLHC];
/* Identifies host controllers. */
struct hostent *hostentry; /* Structure for host name utilities. */

struct tm *timestruct; /* Time structure for TOD. */

/*
 * Gather information on the local host controller and initialize
 * the CONTROLLER_LIST_ENTRY array.
 */

gethostname(hostname, 6); /* Get HOST name. */

mult_addr_ptr = getalladdrs(hostname); /* Get Internet addresses. */
inet_address = mult_addr_ptr->addr_array[0];

dev_addr_ptr = getdevaddrs(inet_address); /* Get controller IDs. */

controller_list[0].controller_number =
    dev_addr_ptr->dev_addr_array[0].dev_addr;

/*
 * Set up the user socket environment. Note: we are NOT using
 * synchronizers, and the socket is NOT privileged.
 */

setup_socket_env(0, dev_addr_ptr->num_devs, controller_list, NULL);

/*
 * Create a socket and bind our local address.
 */
s = socket(AF_INET, SOCK_STREAM, 0); /* Create socket. */

my_name.sin_family = AF_INET; /* Address format. */
my_name.sin_port = 0; /* Let the socket interface
                        select a port for us. */

my_name.sin_addr.s_addr =
    dev_addr_ptr->dev_addr_array[0].internet_address;

bind(s, &my_name, namelen); /* Bind the socket. */

/*
 * Gather information on the server host.
 */

hostentry = gethostbyname("CAL_SVR"); /* Get data on the host. */

rem_name.sin_family = AF_INET; /* Address format. */
rem_name.sin_port = 1101; /* Server port number. */
rem_name.sin_addr.s_addr = *(unsigned long *)hostentry->h_addr;

/*
 * Connect to the server host and request the TOD. Wait for a
 * response and display the data.
 */

```

```
connect(s, &rem_name, namelen);          /* Connect to remote host. */

send(s, req, strlen(req), 0);             /* Send message. */

recv(s, rx_buff, sizeof(rx_buff), 0);    /* Receive TOD. */

timestruct = (struct tm *)rx_buff;        /* Cast to time structure. */

printf("Time is %d:%d:%d on %d-%d-%d.\n",
       timestruct->tm_hour, timestruct->tm_min, timestruct->tm_sec,
       timestruct->tm_mon+1, timestruct->tm_mday, timestruct->tm_year);

close_socket(s);
close_socket_env();
}
```

Sample Program 2: Server Program

The server program performs the following steps:

1. Creates condition handler (on-unit) to handle asynchronous events.
2. Obtains information about the local host.
3. Sets up the socket environment.
4. Creates sockets and listens for connections.
5. When a connection request comes in, accepts the connection and returns the time of day to the client.

Commented Program

```
/*
 * TCP_CALENDAR_SERVER.C:   example of a TCP server process.
 */

/*
 * Include files.
 */

#include <stdio.h>
    /* PRIMOS C I/O header file. */

#include <keys.h>
    /* Keys for ER$PRINT subroutine. */

#include <time.h>
    /* Header file for time() and localtime(). */

#include <socket.h>
    /* Socket-related definitions. */
```



```

#include <ioctl.h>
    /* Definitions for ioctl function. */

#include <in.h>
    /* Internet constants and structures. */

#include <inet.h>
    /* External declarations for network library routines. */

#include <netdb.h>
    /* Structures used by network library routines. */

#include <tcp/ip_time.h>
    /* Time structures from BSD. */

#include <tcp/ip_error.h>
    /* Error code definitions for socket routines. */

#include <redefine.h>
    /* Allows program to omit TUP$ prefix on routine names. */

/*
 * Routine declarations
 */

/* C library function declarations */
extern long    time();           /* Time as an integer. */
extern struct tm *localtime();  /* Time as a structure. */

/* Socket declarations */
extern long    accept();
extern long    bind();
extern long    close_socket();
extern long    close_socket_env();
extern long    ioctl();
extern long    listen();
extern long    select();
extern long    send();
extern long    setup_socket_env();
extern long    socket();

/* Local declarations */
void    sendtimetocaller();
void    sigquit();             /* On-unit. */

/* System subroutine declarations */
fortran mkon$();               /* Create an on-unit. */
fortran ER$print();            /* Subsystem error reporting. */

typedef struct v80 {           /* Structures for ER$print(). */
    short length;
    char  string[80];
} VAR80_STRING;

```

```
typedef struct v32 {
    short length;
    char string[32];
} VAR32_STRING;

/*
 * Start of code
 */

calendar_server()
{
    long i; /* General purpose index. */
    long cc; /* Socket condition code. */
    long s; /* Socket identifier. */
    short length; /* Argument to mkon$. */
    long backlog; /* Backlog for listen call. */
    long reads; /* Argument for select call. */
    long rdmask; /* Read mask for select call. */
    long NBIOOn; /* Argument for ioctl. */
    unsigned long inet_address; /* Internet address. */
    char hostname[6]; /* Host name (PRIME). */
    struct sockaddr_in my_name; /* Socket name. */
    long namelen = sizeof(my_name); /* Name length. */
    struct mult_addrs *mult_addr_ptr; /* Internet address structure. */
    struct dev_addrs *dev_addr_ptr; /* Device address structure. */
    CONTROLLER_LIST_ENTRY controller_list[MAXLHC];
    /* Identifies host controllers. */
    VAR80_STRING my_message; /* Personal error report. */
    VAR32_STRING routine; /* Routine name. */

/*
 * Set up a condition handler for logout, break, sending to a closed
 * socket, and controller failure.
 */

    length = strlen("LOGOUT$");
    mkon$( "LOGOUT$", length, &sigquit$ );
    length = strlen("QUIT$");
    mkon$( "QUIT$", length, &sigquit$ );
    length = strlen("SIGPIPE");
    mkon$( "SIGPIPE", length, &sigquit$ );
    length = strlen("SIGCONTROLLERDOWN");
    mkon$( "SIGCONTROLLERDOWN", length, &sigquit$ );

/*
 * Gather information on the local host controller and initialize the
 * CONTROLLER_LIST_ENTRY array.
 */

    gethostname(hostname, 6); /* Get HOST name. */

    mult_addr_ptr = getalladdrs(hostname); /* Get Internet addresses. */
    inet_address = mult_addr_ptr->addr_array[0];

    dev_addr_ptr = getdevaddrs(inet_address); /* Get controller IDs. */
}
```

```

    for (i = 0; i < dev_addr_ptr->num_devs; i++)
        controller_list[i].controller_number =
            dev_addr_ptr->dev_addr_array[i].dev_addr;

/*
 * Set up the user socket environment. Note: we are NOT using
 * synchronizers.
 */

cc = setup_socket_env(SO_PRIVILEGED, dev_addr_ptr->num_devs,
                     controller_list, NULL);

if (cc == -1)
{
    /* ERRNO has error code. */
    TUP$ perror("calendar server");
    close_socket_env();
    exit(0);
}

/*
 * Create a socket for each controller, bind them and listen.
 */

my_name.sin_family = AF_INET;      /* Address format. */
my_name.sin_port = 1101;           /* We define the PORT. */

for (i = 0; i < dev_addr_ptr->num_devs; i++)
{
    my_name.sin_addr.s_addr =
        dev_addr_ptr->dev_addr_array[i].internet_address;

    s = socket(AF_INET, SOCK_STREAM, 0);    /* Create socket. */

    rdmask |= (1 << s);                    /* READ mask for select. */

    bind(s, &my_name, namelen);            /* Bind the socket. */

    NBIOOn = 1;
    ioctl(s, FIONBIO, (char *)&NBIOOn);    /* Set nonblocking I/O. */

    backlog = 5;                          /* Allow backlog of 5. */
    cc = listen(s, backlog);                /* Wait for client request. */
    if (cc == -1)
    {
        /* ERRNO has the error code. */
        strcpy(my_message.string, "calendar server");
        my_message.length = strlen(my_message.string);
        strcpy(routine.string, "listen");
        routine.length = strlen(routine.string);
        ER$print((short)k$irtn, tcpip_error_subsystem, (short)errno,
                  my_message, routine);
        close_socket(s);
        close_socket_env();
        exit(0);
    }
} /* End "for (i = 0; i < dev_addr_ptr->num_devs; i++)" */

```

```
/*
 * Wait for a connect request from a client. When one comes in,
 * accept the connection and supply the time of day.
 */

for(;;)
{
    /* Run forever. */
    reads = rdmask ;          /* Work with copy of the READ mask. */
    /* Wait indefinitely for incoming data. */
    if (select (dev_addr_ptr->num_devs, &reads, NULL, NULL, NULL) )
    {
        /* We have read activity. */
        for (i = 0; ((1 << i) <= reads) && (i < 32); i++)
        {
            /* Determine what socket had activity. */
            if ((1 << i) & reads)
                sendtimetocaller(i);    /* Send TOD. */
        }
    } /* End "if (select ... )" */
} /* End "for (;)" */
} /* End "calendar_server()" */

/*
 * sendtimetocaller(): Perform user request: accept connection and
 * then send out the time to the caller
 *
 * Argument: s Socket the request is on
 */
void sendtimetocaller(s)
long s ;
{
    long ns;                    /* Accepted socket. */
    long seconds;               /* Argument to time(). */
    struct tm *timestruct;      /* Time structure. */
    long tmstructsize = sizeof(struct tm);
    struct sockaddr_in rem_name; /* Socket name. */
    long namelen = sizeof(rem_name); /* Name length. */

    /* Accept socket and get the TOD. */
    ns = accept(s, &rem_name, &namelen);

    time(&seconds);             /* What time is it??? */
    timestruct = localtime(&seconds); /* Convert to a time structure. */

    /* Send TOD to the requesting client. */

    send(ns, timestruct, tmstructsize, 0);
    close_socket(ns);           /* Close the socket. */
    return ;
} /* End "sendtimetocaller()" */

/*
 * sigquit$: Condition handler.
 */
```

```
void sigquit$(csf)
char *csf;          /* Condition stack frame. */
{
    printf("Received an on-condition\n");
    printf("Closing socket environment ... \n");
    close_socket_env();
    exit(0);
}
```

Part III: Administrator

Installing PRIMOS TCP/IP

This chapter describes

- How a Network Administrator or operator installs PRIMOS TCP/IP
- The files and directories that comprise PRIMOS TCP/IP after software is installed
- How PRIMOS TCP/IP and Network Terminal Service (NTS) software modify ACLs within TCP/IP directories
- How the PRIMOS TCP/IP installation procedure modifies the systemwide ENTRY\$ search rules in order to make the PRIMOS socket library accessible to programmers

Information in this chapter is intended only for Network Administrators or operators.

Figure 11-1 shows the sequence of steps to follow when installing TCP/IP on a 50 Series system. The next section explains these steps.

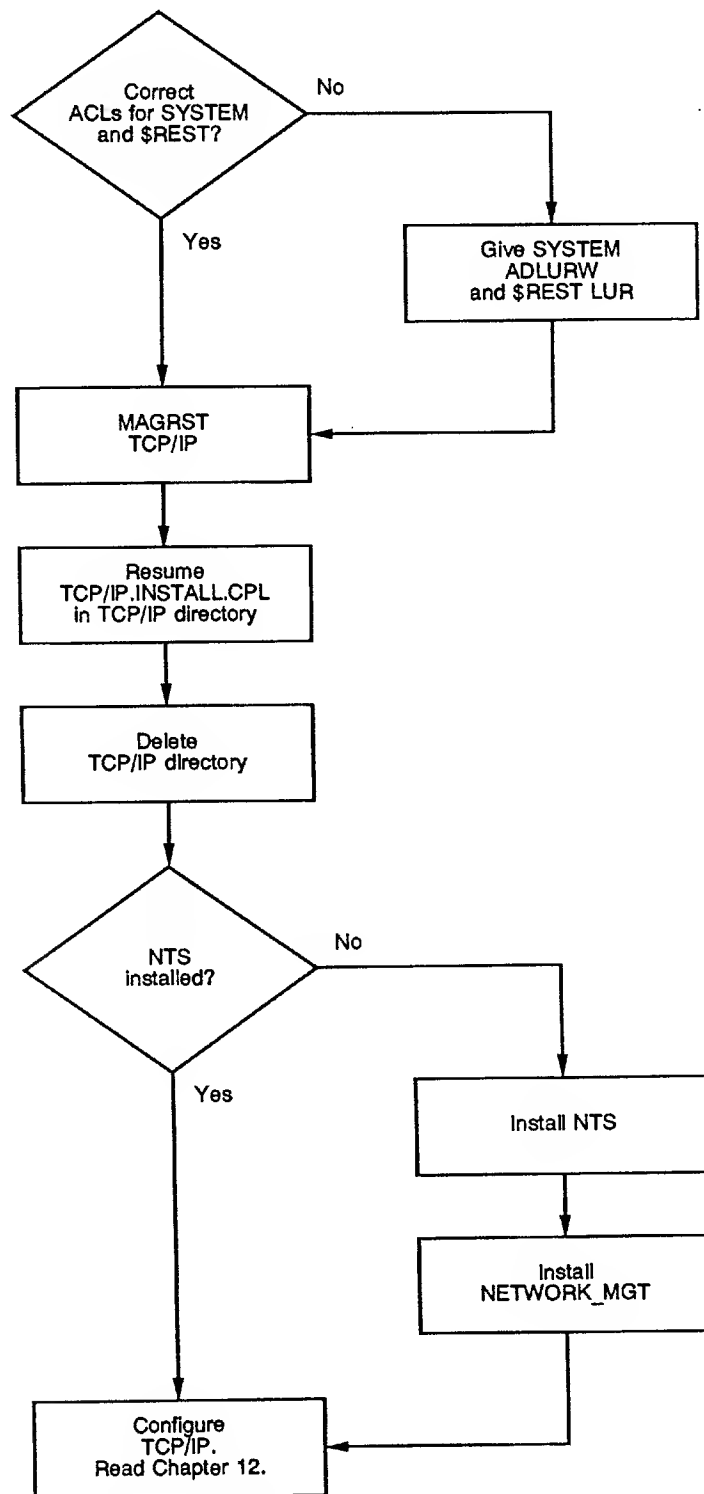
Prerequisites

PRIMOS TCP/IP requires three software products:

- PRIMOS TCP/IP software
- NTS software
- LAN300 Network Management software

NTS is a requirement for the server TELNET program. PRIMOS TCP/IP also shares network management facilities with NTS.

NTS with Network Management is a separate product. PRIMOS TCP/IP is restored from one logical tape; NTS and Network Management are restored from other logical tapes.



Q10155-3LA-4-2

Figure 11-1
PRIMOS TCP/IP Installation Session

Installation Procedure

Use the following procedure to install Release 2.0 PRIMOS TCP/IP software and NTS with Network Management from tape:

1. Log in as SYSTEM.
2. Attach to the Master File Directory (MFD).
3. Use the MAGRST command to restore TCP/IP software from the tape to the partition. MAGRST creates the TCP/IP directory.
4. Attach to the TCP/IP directory. Use the RESUME command to execute the TCP/IP.INSTALL.CPL file. Use the -PRIMENET option if you have PRIMENET installed on your system and you want to send MAIL to other PRIMENET nodes. The default is sending mail via SMTP over a LAN300.

After you install PRIMOS TCP/IP, its files reside in the following directories:

CMDNCO

DOWN_LINE_LOAD*

TCP/IP*

HELP*

INFO

LIB

LIBRARIES*

MAILER*

SYSCOM

SYSOVL

5. Attach to the MFD and delete the TCP/IP directory.
6. Install the Network Terminal Service (NTS) according to the instructions contained in Chapter 2, Installing NTS, of the *NTS Planning and Configuration Guide*.

The last line of the NTS.INSTALL.CPL file invokes the NETWORK_MGT.INSTALL.CPL file, which installs the LAN300 Network Management facility. Refer to Appendix D, NTS-related Configuration Directives, for a description of any special considerations that apply to directives in the system configuration (CONFIG) file when you install NTS for PRIMOS TCP/IP.

Note

ADLURW access and LUR access are the minimum ACL settings required for SYSTEM and \$REST, respectively, at the MFD level on the partitions where TCP/IP-related directories are to be installed.

Sample Session

The following example shows the installation of PRIMOS TCP/IP from a supervisor terminal. You attach to the MFD, check access rights with the LIST_ACCESS (LAC) command, and then assign the magnetic tape drive with the ASSIGN command. After the contents of the tape are restored, you unassign the tape drive with the UNASSIGN command.

In this example, the TCP/IPINSTALL.CPL is executed without the -PRIMENET option. After MAIL is configured and started, messages are delivered only to other nodes on a LAN300.

```
OK, A MFD
OK, LAC

ACL protecting "<Current directory>":
      SYSTEM:  ALL
      $REST:   LUR

OK, ASSIGN MTO
Device MTO assigned.

OK, MAGRST
[MAGRST Rev. 22.0.0 Copyright (c) 1988, Prime Computer, Inc.]
Tape unit (9 Trk): 0
Enter logical tape number:
Name: tcp/ip
Date(MM DD YY): 7-12-88
Rev no: 220
Reel no: 1
Ready to Restore: $I
Ready to Restore: Yes
*** Starting Restore ***
TCP/IP (pasufd)
Password UFD restored as ACL UFD: TCP/IP
TCP/IP>CMDNCO (pasufd)
Password UFD restored as ACL UFD: CMDNCO
TCP/IP>HELP* (pasufd)
Password UFD restored as ACL UFD: HELP*
TCP/IP>DOWN_LINE_LOAD* (pasufd)
Password UFD restored as ACL UFD: DOWN_LINE_LOAD*
.
.
.

*** End Logical tape ***
*** Restore Complete ***
```

OK, **UNASSIGN MTO -UNLOAD**
Device released.

OK, **A TCP/IP**
OK, **R TCP/IP.INSTALL**

TCP/IP Installation started at 88-07-12.14:53:36.Tue

Building a TCP/IP* directory and setting ACLs.

Installing system commands

"TCP/IP>CMDNC0>START_TCP/IP.RUN" copied to "CMDNC0>START_TCP/IP.RUN".
"TCP/IP>CMDNC0>STOP_TCP/IP.RUN" copied to "CMDNC0>STOP_TCP/IP.RUN".
"TCP/IP>CMDNC0>TCP/IP_INIT_HOSTS.RUN" copied to "CMDNC0>TCP/IP_INIT_HOSTS.RUN".
"TCP/IP>CMDNC0>FTP.RUN" copied to "CMDNC0>FTP.RUN".
"TCP/IP>CMDNC0>TELNET.RUN" copied to "CMDNC0>TELNET.RUN".
"TCP/IP>CMDNC0>START_FTP_SERVER.RUN" copied to "CMDNC0>START_FTP_SERVER.RUN".
"TCP/IP>CMDNC0>START_SMTP.RUN" copied to "CMDNC0>START_SMTP.RUN".
"TCP/IP>CMDNC0>START_MAILER.RUN" copied to "CMDNC0>START_MAILER.RUN".
"TCP/IP>CMDNC0>STOP_MAILER.RUN" copied to "CMDNC0>STOP_MAILER.RUN".
"TCP/IP>CMDNC0>MAIL.RUN" copied to "CMDNC0>MAIL.RUN".

Installing libraries

"TCP/IP>LIB>SOCKET.BIN" copied to "LIB>SOCKET.BIN".
"TCP/IP>LIBRARIES*>SOCKET.RUN" copied to "LIBRARIES*>SOCKET.RUN".
"TCP/IP>LIBRARIES*>SOCKET_IX.RUN" copied to "LIBRARIES*>SOCKET_IX.RUN".
"TCP/IP>LIBRARIES*>MAPS>SOCKET.MAP" copied to "LIBRARIES*>MAPS>SOCKET.MAP".
"TCP/IP>LIBRARIES*>MAPS>SOCKET_IX.MAP" copied to "LIBRARIES*>MAPS>SOCKET_IX.MAP".

Installing include files

"TCP/IP>SYSCOM>IN.H.INS.CC" copied to "SYSCOM>IN.H.INS.CC".
"TCP/IP>SYSCOM>INET.H.INS.CC" copied to "SYSCOM>INET.H.INS.CC".
"TCP/IP>SYSCOM>IOCTL.H.INS.CC" copied to "SYSCOM>IOCTL.H.INS.CC".
"TCP/IP>SYSCOM>NETDB.H.INS.CC" copied to "SYSCOM>NETDB.H.INS.CC".
"TCP/IP>SYSCOM>REDEFINE.H.INS.CC" copied to "SYSCOM>REDEFINE.H.INS.CC".
"TCP/IP>SYSCOM>SOCKET.H.INS.CC" copied to "SYSCOM>SOCKET.H.INS.CC".
"TCP/IP>SYSCOM>TCP/IP_ERROR.H.INS.CC" copied to "SYSCOM>TCP/IP_ERROR.H.INS.CC".
"TCP/IP>SYSCOM>TCP/IP_GLOBAL.H.INS.CC" copied to "SYSCOM>TCP/IP_GLOBAL.H.INS.CC".
"TCP/IP>SYSCOM>TCP/IP_TIME.H.INS.CC" copied to "SYSCOM>TCP/IP_TIME.H.INS.CC".

Installing error text file

"TCP/IP>SYSOVL>TUP\$_ERROR_TABLE" copied to "SYSOVL>TUP\$_ERROR_TABLE".

Installing run files

"TCP/IP>TCP/IP*>RUN>TCP/IP_MANAGER.RUN" copied to "<0>TCP/IP*>RUN>TCP/IP_MANAGER.RUN".
"TCP/IP>TCP/IP*>RUN>TCP/IP_DUMP.RUN" copied to "<0>TCP/IP*>RUN>TCP/IP_DUMP.RUN".
"TCP/IP>TCP/IP*>RUN>TCP/IPFTPSP.RUN" copied to "<0>TCP/IP*>RUN>TCP/IPFTPSP.RUN".
"TCP/IP>TCP/IP*>RUN>TCP/IPFTPSS.RUN" copied to "<0>TCP/IP*>RUN>TCP/IPFTPSS.RUN".
"TCP/IP>TCP/IP*>RUN>TCP/IPFTPUP.RUN" copied to "<0>TCP/IP*>RUN>TCP/IPFTPUP.RUN".
"TCP/IP>TCP/IP*>RUN>TCP/IPFTPUS.RUN" copied to "<0>TCP/IP*>RUN>TCP/IPFTPUS.RUN".
"TCP/IP>TCP/IP*>RUN>SMTP.RUN" copied to "<0>TCP/IP*>RUN>SMTP.RUN".

Installing Mail files

```
"TCP/IP>MAILER*>EXEC>MAILER.CPL" copied to "MAILER*>EXEC>MAILER.CPL".
"TCP/IP>MAILER*>EXEC>ISC_MAILER.RUN" copied to "MAILER*>EXEC>MAILER.RUN".
```

Installing miscellaneous TCP/IP files

```
"TCP/IP>TCP/IP*>MESSAGE_TEXT>TCP/IP_MES.OUT" copied to
"<0>TCP/IP*>MESSAGE_TEXT>TCP/IP_MES.OUT".
"TCP/IP>TCP/IP*>MESSAGE_TEXT>TCP/IPHTU_MES.OUT" copied to
"<0>TCP/IP*>MESSAGE_TEXT>TCP/IPHTU_MES.OUT".
"TCP/IP>TCP/IP*>MESSAGE_TEXT>TEL_MES.OUT" copied to
"<0>TCP/IP*>MESSAGE_TEXT>TEL_MES.OUT".
"TCP/IP>TCP/IP*>TCP/IPHTU.RUN" copied to "<0>TCP/IP*>TCP/IPHTU.RUN".
"TCP/IP>TCP/IP*>TCP/IP.INSTALL_ACL.CPL" copied to "<0>TCP/IP*>TCP/IP.INSTALL_ACL.CPL".
"TCP/IP>TCP/IP*>HOSTS.TEMPLATE" copied to "<0>TCP/IP*>HOSTS.TEMPLATE".
"TCP/IP>TCP/IP*>THISHOST.TEMPLATE" copied to "<0>TCP/IP*>THISHOST.TEMPLATE".
"TCP/IP>TCP/IP*>SEMAPHORE_ALLOCATION_FILE" copied to
"<0>TCP/IP*>SEMAPHORE_ALLOCATION_FILE".
```

Installing controller download file

```
"TCP/IP>DOWN_LINE_LOAD*>LHC300_TCP.DL" copied to "DOWN_LINE_LOAD*>LHC300_TCP.DL".
```

Modifying system search rules

```
EDIT
L SOCKET.RUN;D
LIBRARIES*>SOCKET.RUN
B;I LIBRARIES*>SOCKET.RUN
T;L SOCKET_IX.RUN;D
LIBRARIES*>SOCKET_IX.RUN
B;I LIBRARIES*>SOCKET_IX.RUN
file
ENTRY$.SR
```

```
=====
Setting ACLs on the SAD.
=====
```

```
a mfd
edac SAD .tcp_ftp$:lur -nq
a SAD
edac @ .tcp_ftp$:lur -nq
edac *>@>@ .tcp_ftp$:lur -nq
edac UVF .tcp_ftp$:pdalurwx -nq
attach <REV22>SEARCH_RULES*
```

TCP/IP installation successfully completed at 88-06-30.14:55:32.Thu

```
OK, A MFD
OK, DELETE TCP/IP
Ok to delete directory "TCP/IP"? YES
"TCP/IP" deleted.
```

PRIMOS TCP/IP Directories and Files

Network Administrators and operators should be familiar with the directories and files that PRIMOS TCP/IP comprises. The PRIMOS TCP/IP installation program puts TCP/IP software in the following directories:

CMDNCO

DOWN_LINE_LOAD*

TCP/IP*

TCP/IP*>LOG_FILES

TCP/IP*>MESSAGE_TEXT

TCP/IP*>RUN

TCP/IP*>SMTP

HELP*

INFO

LIB

LIBRARIES*

LIBRARIES*>MAPS

MAILER*

MAILER*>DB

MAILER*>EXEC

MAILER*>LOG

MAILER*>MBOX

MAILER*>QUEUE

SYSCOM

SYSOVL

CMDNC0 Directory

After installation, the following runfiles reside in CMDNC0:

FTP.RUN

Enables users to invoke the FTP program.

MAIL.RUN

Enables users to invoke the MAIL program.

START_FTP_SERVER.RUN

Enables you to start the PRIMOS FTP Server process.

START_MAILER.RUN

Enables you to start the MAILER_DAEMON server.

START_SMTP.RUN

Enables you to start the MAIL servers SMTP_SENDER0 and SMTP_SERVER.

START_TCP/IP.RUN

Enables you to start PRIMOS TCP/IP protocols on the LHC.

STOP_MAILER.RUN

Enables you to stop the MAILER_DAEMON server.

STOP_TCP/IP.RUN

Enables you to stop all TCP/IP-related processes.

TELNET.RUN

Enables users to invoke the user TELNET program.

DOWN_LINE_LOAD* Directory

After installation, the following file resides in the DOWN_LINE_LOAD* directory:

LHC300_TCP.DL

PRIMOS TCP/IP protocols.

TCP/IP* Directory

After installation, the following files reside in the TCP/IP* directory:

HOSTS.TEMPLATE

A template of the HOSTS.TXT configuration file.

THISHOST.TEMPLATE

A template of the THISHOST configuration file.

SEMAPHORE_ALLOCATION_FILE

A file required by PRIMOS TCP/IP.

TCP/IP.INSTALL_ACL.CPL

A CPL file executed as part of the PRIMOS TCP/IP installation procedure to set ACLs on the System Administration Directory (SAD).

TCP/IPHTU.RUN

The runfile for invoking the Host Table Utility.

TCP/IP*>COMI Directory

The installation procedure creates TCP/IP*>COMI, an empty directory.

TCP/IP*>ISC Directory

The installation procedure creates TCP/IP*>ISC, an empty directory.

TCP/IP*>LOG_FILES Directory

After installation, the following empty directory resides in the TCP/IP*>LOG_FILES directory:

ARCHIVE

Contains the log files created prior to the last time PRIMOS TCP/IP was stopped and restarted.

TCP/IP*>MESSAGE_TEXT Directory

After installation, the following files reside in the TCP/IP*>MESSAGE_TEXT directory:

TCP/IPHTU_MES.OUT

Message file for the Host Table Utility (TCP/IPHTU).

TCP/IP_MES.OUT

Message file for other TCP/IP software.

TEL_MES.OUT

Message file for TELNET.

TCP/IP*>RUN Directory

After installation, the following files reside in the TCP/IP*>RUN directory:

SMTP.RUN

Runfile for the SMTP processes.

TCP/IPFTPSP.RUN

The Server process phantom runfile.

TCP/IPFTPSS.RUN

The FTP server process runfile.

TCP/IP_DUMP.RUN

The runfile for a diagnostic tool for your customer service representative.

TCP/IP_MANAGER.RUN

The PRIMOS TCP/IP Manager process runfile.

TCP/IP*>SMTP Directory

After installation, the following subdirectories reside in the TCP/IP*>SMTP directory:

INCOMING

The queue for incoming messages.

OUTGOING

The queue for outgoing messages.

HELP* Directory

After installation, the following files reside in the HELP* directory:

FTP.HELP

HELP file for PRIMOS FTP.

MAIL.HELP

HELP file for TCP/IP MAIL.

START_FTP_SERVER.HELP

HELP file for the START_FTP_SERVER command.

START_MAILER.HELP

HELP file for the START_MAILER command.

START_SMTP.HELP

HELP file for the START_SMTP command.

START_TCP/IP.HELP

HELP file for the START_TCP/IP command.

STOP_MAILER.HELP

HELP file for the STOP_MAILER command.

STOP_TCP/IP.HELP

HELP file for TCP/IP STOP commands.

TCP/IPHTU.HELP

HELP file for the TCP/IP Host Table Utility (TCPHTU).

TELNET.HELP

HELP file for the user TELNET facility.

INFO Directory

After installation, the following file resides in the INFO directory:

TCP/IP.RUNO

The PRIMOS TCP/IP RUNOFF output INFO file.

LIB Directory

After installation, the following file resides in the LIB directory:

SOCKET.BIN

The socket library binary file.

LIBRARIES* Directory

After installation, the following files reside in the LIBRARIES* directory:

SOCKET.RUN

The V-mode socket library entrypoint runfile.

SOCKET_IX.RUN

The IX-mode socket library entrypoint runfile.

LIBRARIES*>MAPS Directory

After installation, the following files reside in the LIBRARIES*>MAPS directory:

SOCKET.MAP

The BIND map of the SOCKET.RUN file.

SOCKET_IX.MAP

The BIND map of the SOCKET_IX.RUN file.

MAILER*>DB Directory

After installation, the following files reside in the MAILER*>DB directory:

CONFIGURATION.TEMPLATE

A template of the CONFIGURATION file.

MAILER*>EXEC Directory

After installation, the following files reside in the MAILER*>EXEC directory:

MAILER.CPL

A CPL file that starts the MAILER_DAEMON.

START_MAILER.RUN

Enables you to start MAILER_DAEMON and the MAIL servers.

STOP_MAILER.RUN

Enables you to stop the MAILER_DAEMON.

MAILER*>LOG Directory

The MAILER*>LOG directory is an empty directory that contains MAIL log files after the MAIL servers start.

MAILER*>MBOX Directory

The MAILER*>MBOX directory is an empty directory that contains user mailboxes after the MAIL servers start and after users initialize their mailboxes. If you reinstall TCP/IP (including MAIL), current mailboxes and their contents are preserved.

MAILER*>QUEUE Directory

The MAILER*>QUEUE directory is an empty directory that contains the input and output MAIL queues after the MAIL servers start.

SYSCOM Directory

After installation, the following socket library include files reside in the SYSCOM directory:

IN.H.INS.CC

Declarations of Internet constants and structures.

INET.H.INS.CC

External declarations for network library routines.

IOCTL.H.INS.CC

Definitions of two socket options.

NETDB.H.INS.CC

Declarations of structures returned by the network library subroutines.

REDEFINE.H.INS.CC

#define statements that replace the customary UNIX routine names with the corresponding PRIMOS routine names.

SOCKET.H.INS.CC

Socket-related definitions and declarations.

TCP/IP_ERROR.H.INS.CC

Definitions of socket errors.

TCP/IP_GLOBAL.H.INS.CC

Global declarations and definitions.

TCP/IP_TIME.H.INS.CC

Declarations of structures relating to time functions.

SYSOVL Directory

After installation, the following file resides in the SYSOVL directory:

TUP\$_ERROR_TABLE

Socket library error messages.

Access Control List Settings

This section describes how PRIMOS TCP/IP and supporting software installation utilities modify ACLs within TCP/IP-related directories.

The diagrams in the next sections show what ACL settings are required to successfully use PRIMOS TCP/IP.

ACLs Modified by Installing PRIMOS TCP/IP

After you start PRIMOS TCP/IP, the PRIMOS FTP server and the phantoms that it spawns belong to the .TCP_FTP\$ ACL group. The FTP server must access the SAD to verify user IDs, passwords, and project IDs (if required).

TCP/IP*>TCP/IP.INSTALL_ACL.CPL, which the TCP/IP.INSTALL.CPL file invokes automatically, sets access rights to the SAD. If ACLs on the SAD are changed for any reason, PRIMOS TCP/IP may not work correctly. For example, if the User Profile Database is rebuilt with the EDIT_PROFILE REBUILD command, you must manually reset the ACLs on the SAD, its subdirectories, and its files. To reset ACLs on the SAD, you must run TCP/IP.INSTALL_ACL.CPL again.

Note

If you use EDIT_PROFILE to create a new project for remote users after you install TCP/IP, you must run the TCP/IP.INSTALL_ACL.CPL file to reset ACLs on the SAD. Chapter 14, Activating PRIMOS TCP/IP, describes how you start server FTP so that it prompts users to enter a project ID.

Figure 11-2 shows ACLs set on the SAD and its files.

```

SAD - .TCP_FTP$:LUR
|
|
| MGF - .TCP_FTP$:LUR
| MPF - .TCP_FTP$:LUR
| SDF - .TCP_FTP$:LUR
| UVF - .TCP_FTP$:PDALURWX
|
-----
|                                     |
| DEFAULT - .TCP_FTP$:LUR             | project_name - .TCP_FTP$:LUR
|                                     |
| MPP - .TCP_FTP$:LUR                 | MPP - .TCP_FTP$:LUR
| PDF - .TCP_FTP$:LUR                 | PDF - .TCP_FTP$:LUR
| PPPF - .TCP_FTP$:LUR                | PPPF - .TCP_FTP$:LUR
| PVF - .TCP_FTP$:LUR                 | PVF - .TCP_FTP$:LUR

```

Figure 11-2
ACLs Set on the SAD

Figure 11-3 shows the ACLs set on the TCP/IP* directory and four subdirectories by the TCP/IP>TCP/IP.INSTALL.CPL file.

TCP/IP*					
					DSMASR:PDALURWX
					DSMNETSR:LUR
					DSMSR:LUR
					SYSTEM:PDALURWX
					.NETWORK_MGT\$:LUR
					.TCP/IP\$:PDALURWX
					\$REST:LUR

COMI	RUN	MESSAGE_TEXT	LOG_FILES	ISC	SMTP

DSMASR:PDALURWX			DSMASR:PDALURWX		
DSMNETSR:LUR			DSMNETSR:LUR		
DSMSR:LUR			DSMSR:LUR		
SYSTEM:PDALURWX			SYSTEM:PDALURWX		
.NETWORK_MGT\$:LUR			.NETWORK_MGT\$:LUR		
.TCP/IP\$:PDALURWX			.TCP/IP\$:PDALURWX		
\$REST:LUR			\$REST:NONE		

Figure 11-3
ACLs Set on the TCP/IP* Directory

ACLs Modified by Installing NTS

Figure 11-4 shows the ACLs modified when NTS is installed.

DOWN_LINE_LOAD*		LHC_DLL_SERVER:LUR
		LTS_DLL_SERVER:LUR
UP_LINE_DUMP*		LHC_ULD_SERVER:ALL
		LTS_ULD_SERVER:ALL
NTS*		SYSTEM:ALL
		\$REST:LUR

Figure 11-4
ACLs Set by NTS

Figure 11-5 shows the ACLs on the NETWORK_MGT* directory after NTS is installed.

```
NETWORK_MGT* - DSM_LOGGER:ALL
              |      SYSTEM:ALL
              |      $REST:LUR
```

Figure 11-5
ACLs on the NETWORK_MGT* Directory

Maintaining System Search Rules

The PRIMOS TCP/IP installation procedure modifies the systemwide ENTRY\$ search rules in order to make the PRIMOS socket library accessible to programmers. If these search rules are changed for any reason, perform the following steps:

1. Attach to the system's SEARCH_RULES* directory.
2. Edit the file ENTRY\$.SR, adding the files LIBRARIES*>SOCKET.RUN and LIBRARIES*>SOCKET_IX.RUN to the list of search rules.
3. Issue the SET_SEARCH_RULES command as follows:

```
OK, SET_SEARCH_RULES ENTRY$.SR
```


Configuring PRIMOS TCP/IP

This chapter explains how a Network Administrator or operator configures PRIMOS TCP/IP. The chapter begins with an overview of the configuration process. A description of Internet address formats precedes a description of the configuration files. To create the required configuration files, you must be familiar with the format of Internet addresses. PRIMOS TCP/IP supports wildcard gateways and subnets. These features are also described.

Chapter 13 describes how to configure TCP/IP MAIL.

Overview

To configure PRIMOS TCP/IP, perform the following tasks:

- Create three configuration files with a text editor
- Run the PRIMOS TCP/IP Host Table Utility (TCP/IPHTU) to format the configuration files
- Modify the system configuration file
- Configure the Network Terminal Service (NTS) for TELNET
- Configure Distributed Systems Management (DSM) to receive unsolicited messages from PRIMOS TCP/IP

Figure 12-1 shows the sequence of steps to configure PRIMOS TCP/IP. The sections that follow explain these steps in detail.

Configuring PRIMOS TCP/IP

PRIMOS TCP/IP requires the following three configuration files:

HOSTS.TXT

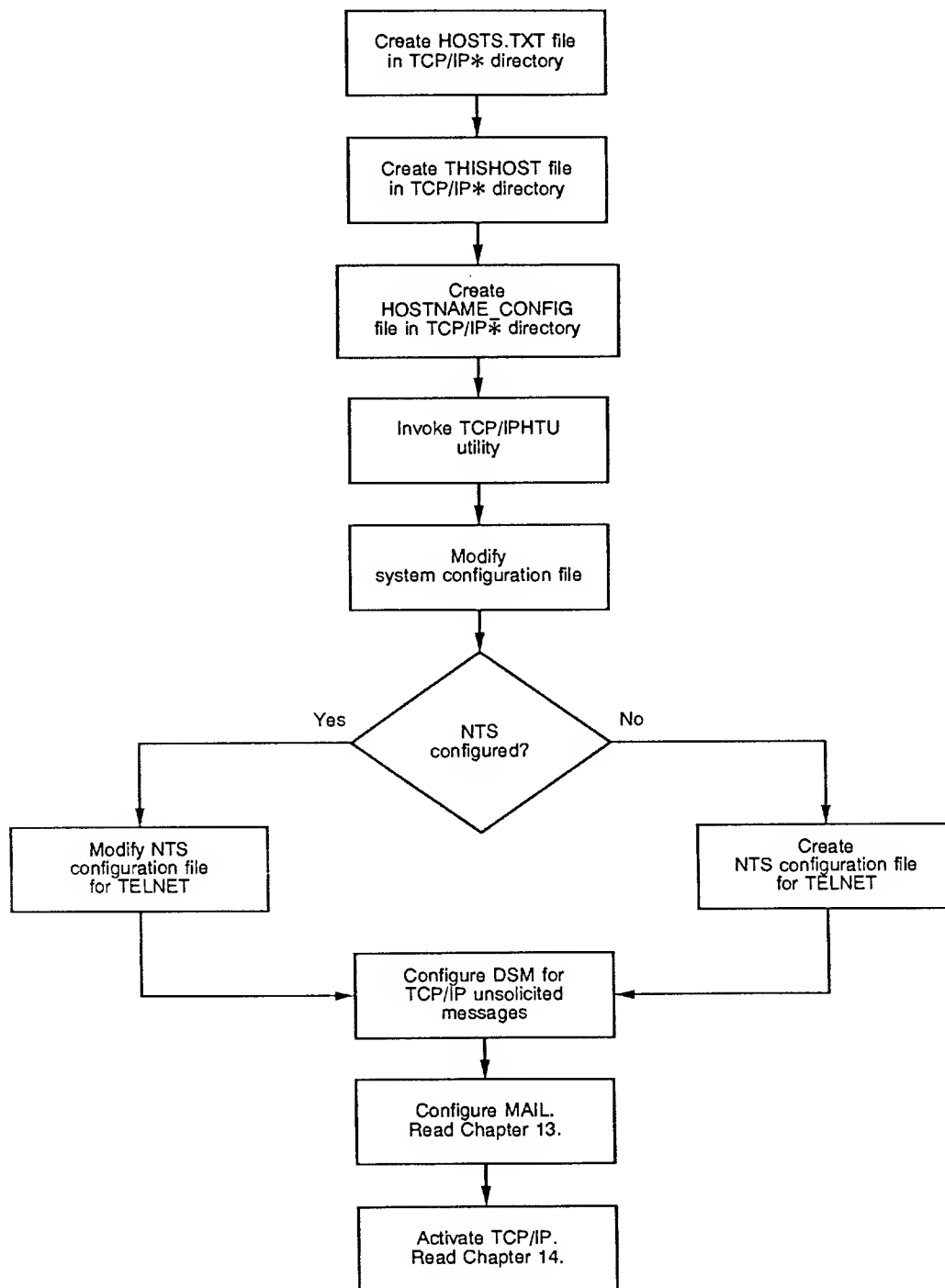
Contains information about the local 50 Series system, its network and the other hosts, gateways, and networks to which the local system can connect.

THISHOST

Identifies the 50 Series system as the local host.

HOSTNAME_CONFIG

Describes information that the Internet Hostname Service protocol requires.



Q10155-3LA-5-2

Figure 12-1
PRIMOS TCP/IP Configuration Session

You create these configuration files with a standard Prime text editor such as ED or EMACS. All configuration files reside in the TCP/IP* directory.

The next section describes how to create Internet addresses that describe the networks and hosts in the configuration files.

Internet Addresses

An Internet address contains both a network number and a local host number. The address is 32 bits long and is divided into four octets (8-bit fields). Each octet can be represented by a decimal number and is separated by a period: for example, 192.12.77.14. The convention of separating the decimal numbers by periods is referred to as dot notation.

Internet addresses have three address formats (Classes A, B, and C). You specify the format in one or all of the three most significant bits of the first octet. (See RFC 796, "Address Mappings," in the *DDN Protocol Handbook* for a definition and complete description.)

Class A: The Class A format uses the first octet for the network number. Bit one of the first octet is 0. Class A format provides seven bits for the network number and 24 bits for the local address. Network numbers range from 1 through 126, inclusive. (Do not use 0 or 127 as a network number. See the section entitled Selecting an Address for an explanation.) The local host number consists of three octets (24 bits). The possible range of values in decimal for each octet of the host number is 1 through 254. Many Class A network numbers are allocated to large, non-commercial networks. The ARPANET and MILNET are Class A networks.

Figure 12-2 illustrates the coding for a Class A address.

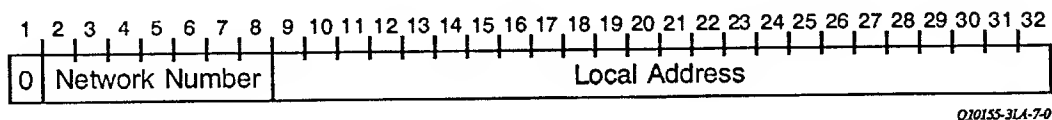
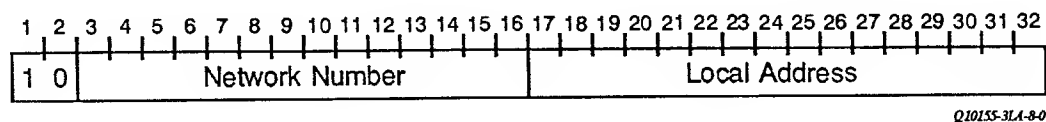


Figure 12-2
Class A Address Format

Class B: The Class B format uses the first two octets for the network number. Bits one and two of the first octet are 1 and 0, respectively. Class B format provides 14 bits for the network number and 16 bits for the local address. Network numbers range from 128.1 through 191.254, inclusive. (Do not use 0 or 255 as network numbers. See the section entitled Selecting an Address for an explanation.) A local address of 16 bits allows a possible 64,536 (2^{16}) computers to be attached to the network. The possible range (in decimal) for the bytes of the local address portion of the address is 1 through 64,536. Class B address formats are generally used on IEEE 802.3 compliant LANs.

Figure 12-3 illustrates the coding for a Class B address.

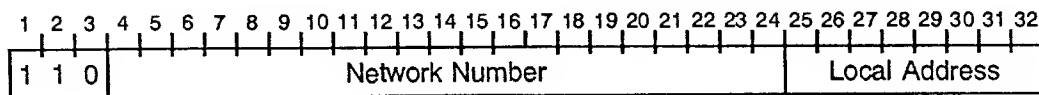


Q10155-31A-9-0

Figure 12-3
Class B Address Format

Class C: The Class C format uses the first three octets for the network number. Bits one, two, and three of the first octet are 1, 1, and 0, respectively. Class C format provides 21 bits for the network number and only eight bits for the local address. This means that a very large number of network numbers is possible (2^{21}) but you are limited to only 254 host addresses. Network numbers range from 192.1.1 through 223.255.254, inclusive. (Addresses above 223 are reserved for future use.) Host addresses range from 1 through 254.

Figure 12-4 illustrates the coding for a Class C address.



Q10155-31A-9-0

Figure 12-4
Class C Address Format

Selecting an Address: PRIMOS TCP/IP supports Classes A, B, and C Internet address formats. The format of the address that you choose does not matter unless you need to connect to networks that the DDN Network Information Center (NIC) administers. Using one address format rather than another provides no performance advantage. When you select a network address, follow these guidelines:

- If your system is not connected to an NIC-administered network (no gateways), you can select any type of address (Class A, B, or C). The network number of each system must be the same and the local address of each system must be unique.
- If your system is connected to an NIC-administered network, the network number must be an assigned number. All systems on that network must have the same network number. The local address of each system must be unique.
- Avoid using the network numbers 0, 127, or 255. 0 is reserved for systems that do not know the network number of their destinations. (See the section entitled Wildcard Gateways for an explanation of the network number 0.) 127 is a reserved number for systems connected to NIC-administered networks. 255 is used to broadcast a message

to every system on the network. Do not use 0 or 255 even if your system is not connected to an NIC-administered network. Do not select a network number the first octet of which is greater than 223.

The HOSTS.TXT Configuration File

When you configure PRIMOS TCP/IP, you must supply it with the following information:

- Name of the local 50 Series host and its Internet address or addresses
- Name and network number of the local network
- Names and addresses of other hosts on the local network
- Names and addresses of gateways to other networks
- Names and network numbers of other networks
- Names and addresses of hosts on other networks

Put this information into a configuration file that you create with a standard Prime text editor such as ED or EMACS. The PRIMOS TCP/IP configuration file has a standard format. This section describes the format of that file and specific line entries.

Notes

The DDN Network Information Center (NIC) at SRI International distributes a file (Internet Host Table) in the standard format containing information to configure ARPANET, MILNET, and other connected networks. If you are running PRIMOS TCP/IP and connecting to one of these networks, you can use the NIC file. Edit the NIC file to add other hosts, gateways, or networks that it does not include. Delete the hosts with which your system will not communicate.

For a description of the standard file format and how to obtain it, refer to "DoD Internet Host Table Specification," RFC 952, in the *DDN Protocol Handbook — Volume Three*.

Sample HOSTS.TXT Configuration File

The following example shows a PRIMOS TCP/IP configuration file that you can modify with a text editor such as ED or EMACS:

```
NET : 10.0.0.0 : EXP-NET :
NET : 51.0.0.0 : CENTRAL-NET :
NET : 128.1.0.0 : BRANCH-OFFICE :
NET : 192.1.0.0 : LAB-NET :
GATEWAY : 51.0.0.90, 128.1.0.1 : G1-GW : XYZ-500 : XYZ/OP : IP :
```

```
;GATEWAY : 51.0.0.92, 192.1.0.78 : G2-GW : XYZ-600 : XYZ/OP: TCP/IP :
GATEWAY : 51.0.0.93, 192.1.0.12 : G3-GW : GBOX-5000 : OS/GBOX : :
GATEWAY : 51.0.0.94, 10.0.0.5 : G4-GW : GBOX-5000 : OS/GBOX : :
GATEWAY : 128.1.0.2, 10.0.0.6 : G5-GW : GBOX-5000 : OS/GBOX : :
GATEWAY : 192.1.0.12, 10.0.0.7 : G6-GW : XYZ-600 : XYZ/OS : :
GATEWAY : 192.1.0.6, 128.1.4.1 : TEST-GW : GBOX-5000 : OS/GBOX : :
HOST : 51.0.0.61 : PRIME-9955.PRIME.COM, PRIME-A : 9955 : PRIMOS : TCP/IP/SMTP :
HOST : 51.0.0.63 : PRIME-9755.PRIME.COM, PRIME-B : 9755 : PRIMOS : TCP/IP/SMTP :
HOST : 51.0.0.64 : PRIME-PXCL : PXCL: UNIX : TCP/IP :
HOST : 128.1.0.2 : G5 : GBOX-5000 : OS/GBOX : :
HOST : 128.1.0.32 : WS-1.PRIME.COM, WS-1 : SUN : UNIX : TCP/IP :
HOST : 128.1.0.33 : WS-2.PRIME.COM, WS-2 : PXCL : UNIX : TCP/IP :
HOST : 128.1.0.34 : PRIME-9750 : 9750 : PRIMOS : TCP/IP :
HOST : 128.1.0.35 : PRIME-EXL : EXL: UNIX : TCP/IP :
HOST : 192.1.0.4 : WSLAB1.PRIME.COM, WSLAB1 : PCXL : UNIX : TCP/IP :
HOST : 192.1.0.7 : WSLAB2.PRIME.COM, WSLAB2 : EXL : UNIX : TCP/IP :
HOST : 192.1.0.10, 10.3.2.1 : PRIME-9750 : 9750 : PRIMOS : TCP/IP :
HOST : 10.0.2.3 : EOK.PRIME.COM, EOK : 6550 : PRIMOS: TCP/IP/SMTP :
```

Meaning of Line Entries

A line in the configuration file has one of the following formats for networks (NET), gateways (GATEWAY), and hosts (HOST). The meanings of the line entries are explained in the sections that follow.

```
NET : NET-ADDR : NETNAME :
GATEWAY : ADDR, ADDR : NAME : CPUTYPE : OPSYS : PROTOCOLS :
HOST : ADDR, ALTERNATE-ADDR : HOSTNAME, NICKNAME : CPUTYPE :
      OPSYS : PROTOCOLS :
```

NET, GATEWAY, and HOST: NET, GATEWAY, and HOST are keywords introducing a network, a gateway, and a host entry, respectively.

NET-ADDR, ADDR, and ALTERNATE-ADDR: NET-ADDR, ADDR, and ALTERNATE-ADDR are 32-bit Internet addresses in decimal. In the above example, addresses are four decimal numbers separated by periods, with each decimal number representing one octet.

The Internet address for the 50 Series system refers to the LHC controller. If a 50 Series system has two LHCs, use the address (ADDR) for the first controller and the alternate address (ALTERNATE-ADDR) for the second controller, as in the following line:

```
HOST : 192.1.0.10, 10.3.2.1 : PRIME-9750 : 9750 : PRIMOS : TCP/IP :
```

Note

Do not enter an Internet address with leading zeros. For example, entering address 192.1.0.10 as 192.001.000.010 may cause TCP/IP to interpret the address incorrectly and to open a connection to the wrong host.

The two address entries for gateway systems are the Internet addresses of the two networks that the gateway system connects. In the following example, system G1-GW connects the networks CENTRAL-NET and BRANCH-OFFICE:

```
GATEWAY : 51.0.0.90, 128.1.0.1 : G1-GW : XYZ-500 : XYZ/OP : IP :
```

See the section Wildcard Gateways, later in this chapter, for more information on addresses for gateway systems.

NETNAME, NAME, HOSTNAME, and NICKNAME: NETNAME, NAME, HOSTNAME, and NICKNAME are text strings with a maximum of 24 characters that can be alphabetic (A through Z), numeric (0 through 9), the minus sign (-), or the period (.). Periods separate a host name from a domain name that MAIL requires. If you want to send mail to a system, you must enter its domain name. Domain names are described in Chapter 13, Configuring PRIMOS TCP/IP MAIL.

You can enter alphabetic characters in uppercase, lowercase, or a combination of both. The first character of a name must be alphabetic and the last character must not be a minus sign or a period. Names cannot contain blanks or spaces. You cannot use single-character names.

You can use a nickname (or alias) only when you provide a full HOSTNAME. In the following line entry PRIME-9955.PRIME.COM is a domain name that is the full HOSTNAME. PRIME-A is the alias. When a system has a domain name, that name must be listed first.

```
HOST : 51.0.0.61 : PRIME-9955.PRIME.COM, PRIME-A : 9955 : PRIMOS : TCP/IP/SMTP :
```

Use -GATEWAY or -GW as part of the name of a host that serves as an Internet gateway.

If local users send files with FTP to a gateway system or log in to a gateway system with TELNET, that system must have a HOST entry in addition to a GATEWAY entry. In the above sample configuration file, system G5 has both a HOST and a GATEWAY entry. The HOST entry for G5 omits the -GW suffix.

Notes

To avoid confusion, the host name for a 50 Series system in the PRIMOS TCP/IP configuration file should be the same as the PRIMOS system name issued at cold start.

PRIMOS and DoD, however, have different rules for forming host names. DoD supports longer names. PRIMOS system names can include some characters (including & and \$) that are not permitted in DoD names. The period is widely used in PRIMOS system names but has a special meaning in DoD names as a delimiter of domain names. Therefore, a Prime host name should not contain a period. For example, EN.D1 is not a valid name under the DoD standard. EN-D1 is a valid name.

Note

It may not be possible to match names if PRIMOS TCP/IP is installed on a system that already has a PRIMOS name that is not valid under the DoD standard. You can either change the PRIMOS name or assign another name that meets DoD standards to the 50 Series host in the configuration file. If the 50 Series host has two names, instruct users to use the PRIMOS system name for PRIMENET (remote login), NETLINK, and NTS terminal connections and the PRIMOS TCP/IP name for FTP and TELNET connections.

CPUTYPE: CPUTYPE is the machine type, such as 9955 or PC.

OPSYS: OPSYS is the operating system, such as PRIMOS, DOS, or UNIX.

PROTOCOLS: PROTOCOLS are higher-level communication protocols such as NCP/FTP, TCP/TELNET, or TCP/FTP running on HOST or GATEWAY.

Colon: : (colon) is the field delimiter.

Double Colon: :: (double colon) indicates a null field. The following line omits the PROTOCOLS parameter:

```
GATEWAY : 192.1.0.6, 128.1.4.1 : TEST-GW : GBOX-5000 : OS/GBOX : :
```

If you omit any information from a line of the HOSTS.TXT file, you must insert colons. For example, the entry

```
HOST : 51.0.0.61 : PRIME-CENTRAL-A, PRIME-A : 9955 : PRIMOS : TCP/IP :
```

can be represented as

```
HOST : 51.0.0.61 : PRIME-CENTRAL-A, PRIME-A : : : :
```

if you omit CPUTYPE, OPSYS and PROTOCOLS entries. (PRIMOS TCP/IP does not check the validity of the CPUTYPE, OPSYS, and PROTOCOLS entries.) When you omit information, you must insert a space between each colon.

Omitting colons, which the Host Table Utility (TCP/IPHTU) counts to determine the end of a line, causes TCP/IPHTU to fail. Each line entry must have the correct number of colons: NET (3), GATEWAY (6), and HOST (6). Line entries must always end with a colon.

Semicolon: ; (semicolon) indicates that the statement that follows is a comment. The following line is a comment:

```
;GATEWAY : 51.0.0.92, 192.1.0.78 : G2-GW : XYZ-600 : XYZ/OP: TCP/IP :
```

Comma: , (comma) separates an address (ADDR) from an alternate address (ALTERNATE-ADDR).

Wildcard Gateways

PRIMOS TCP/IP supports wildcard gateways. A **wildcard gateway** is a gateway that may route data to either a final destination or another gateway when the address of the gateways and networks or subnetworks on the destination route are not known.

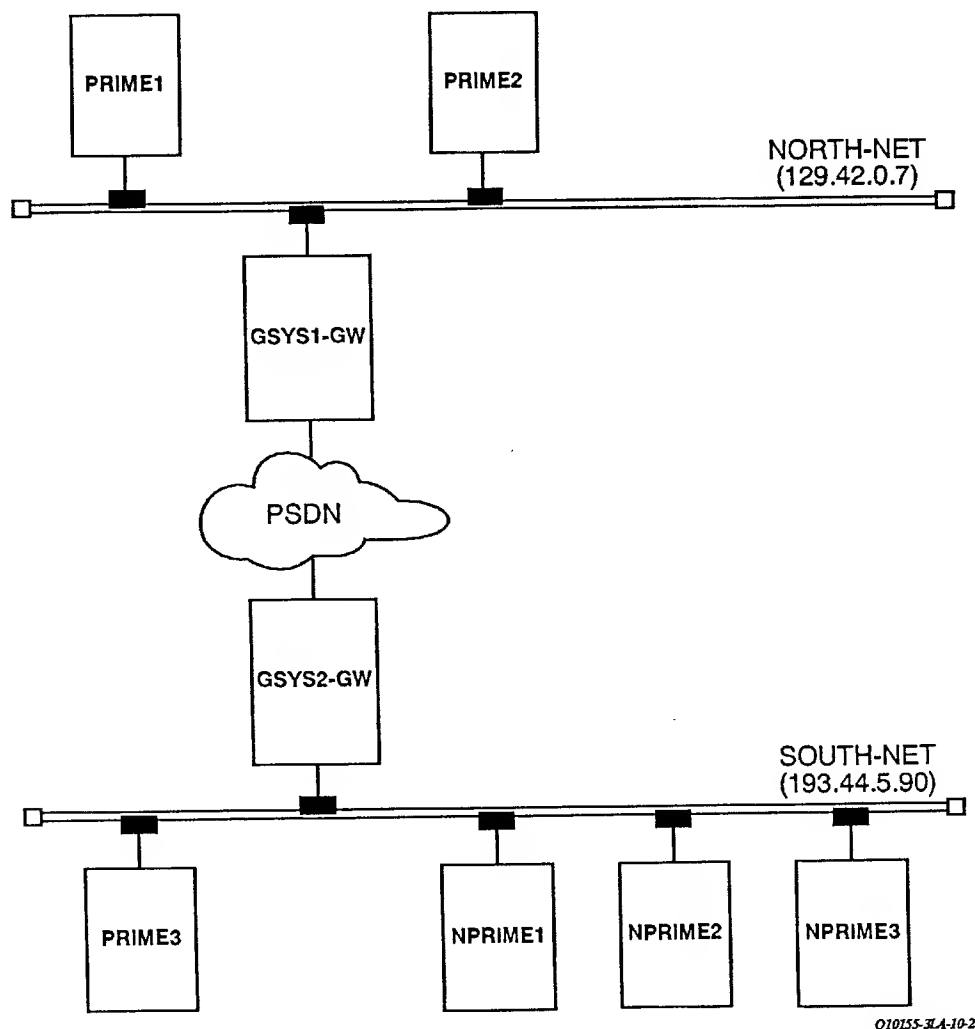
When a user wants to open a connection and transfer a file to a remote system on a remote network, PRIMOS TCP/IP examines tables created from gateway and network entries in the HOSTS.TXT file to construct a route. If the destination can be reached only through many gateways or is part of a subnet, the HOSTS.TXT file ordinarily requires an entry for every remote network or subnet and the gateway to that network or subnet. In some instances, the HOSTS.TXT file requires a very large number of entries. An entry for a wildcard gateway can reduce the number of entries.

When PRIMOS TCP/IP is requested to send data to a system on a remote network, it first looks for a gateway system associated with the address of the remote system and its network. When a gateway receives an Internet Protocol (IP) datagram from PRIMOS TCP/IP, it checks to see whether it knows the destination system, or its gateway and network. If the gateway recognizes the destination address, it forwards the IP datagram to that address. If the gateway recognizes that another gateway provides a better route, it informs PRIMOS TCP/IP. PRIMOS TCP/IP then sends packets to that gateway.

If PRIMOS TCP/IP does not find a gateway for the remote network, it looks for a gateway to network 0. Remember that 0 is not normally a valid network number. PRIMOS TCP/IP always associates a gateway address of 0 with the address of any remote system and network for which a specific gateway is not listed in the HOSTS.TXT file. This special gateway to network 0 is the wildcard gateway.

An Example of a Wildcard Entry in HOSTS.TXT

Figure 12-5 shows 50 Series systems on two networks (NORTH-NET and SOUTH-NET) that are connected across a **Packet Switching Data Network (PSDN)** by two gateway systems (GSYS1-GW and GSYS2-GW). NORTH-NET has two 50 Series systems connected to it. SOUTH-NET has one 50 Series system connected to it.



Q10155-3A-10-2

Figure 12-5
Networks Connected by Wildcard Gateways

The configuration file has two wildcard gateway entries. Each GATEWAY entry consists of two numbers. The first number (0.0.0.0) is the wildcard entry. The second number is the local address of the network to which the gateway is attached. The configuration file of the three 50 Series systems has the following entries:

```

NET : 129.42.0.0 : NORTH-NET :
NET : 193.44.5.0 : SOUTH-NET :
GATEWAY : 0.0.0.0, 129.42.0.7 : GSYS1-GW : XYZ-600 : XYZ/OS : :
GATEWAY : 0.0.0.0, 193.44.5.90 : GSYS2-GW : XYZ-600 : XYZ/OS : :
HOST : 129.42.0.1 : PRIME1 : 9955 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 129.42.0.2 : PRIME2 : 9750 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 193.44.5.1 : PRIME3 : 9955 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 193.44.5.2 : NPRIME1 : WHIZTRONICS 100 : WHIZ-OP : TCP/FTP, TCP/TELNET :
HOST : 193.44.5.3 : NPRIME2 : WS-1200 : WS-OP : TCP/FTP, TCP/TELNET :
HOST : 193.44.5.4 : NPRIME3 : WS-1400 : WS-OP : TCP/FTP, TCP/TELNET :

```

Summary

When you create the HOSTS.TXT configuration file, keep the following points in mind:

- The local host name must appear in the file, along with the names of all other systems with which you may want to communicate.
- You must enter the domain names of systems to which you want to send mail.
- You can choose network names and network numbers arbitrarily as long as there is no possibility of connecting to DDN NIC-administered networks. If that possibility exists, apply to the NIC for an assigned network number. In most cases, connecting to DDN NIC-administered networks is not an issue and network numbers and Internet addresses can be assigned arbitrarily.
- You can enter information in the configuration file in uppercase or lowercase or some combination of both. When the table is formatted by the Host Table Utility, described in a later section, all entries are converted to lowercase.
- Name the configuration file HOSTS.TXT and put it in the TCP/IP* directory.

The THISHOST File

In addition to the HOSTS.TXT file, you must create another file (named THISHOST) that describes the local Prime host. You have listed the name of the local system in the HOSTS.TXT file with other hosts on the network, but you have not specified it as the local host. That is the purpose of the THISHOST file, which contains the host name and information describing the LHCs. Also, if your network supports subnetting (subnets), you enter the subnet mask in the THISHOST file. Subnetting is described in this section.

Format

To specify information about the local system, perform the following steps:

1. Create a file named THISHOST in the TCP/IP* directory using a standard Prime editor such as ED or EMACS.
2. On a single line, enter, in uppercase or lowercase, one of the names by which the local 50 Series host is known. This name must be one of the names by which you identify the host in the HOSTS.TXT file. Enter only one host name on this line. Host names must conform to the DoD standard.
3. For each LHC in use, enter a separate line describing the physical device address of the controller and the controller's Internet address. (You can use the STATUS COMM command to display the device addresses in octal of the LHCs attached to your system. See the *Operator's Guide to System Commands*.)

Specify this information in one or more lines that take the following form:

```
device address = 53 : internet address = 129.5.42.3 :
```

In the above example, the Internet address of controller 53₈ is 129.5.42.3. If the host also uses another controller, the file would consist of three lines.

The following THISHOST file describes one host named SYS6 with two controllers:

```
SYS6
device address = 53 : internet address = 129.5.42.3 :
device address = 54 : internet address = 10.2.3.4 :
```

Summary: Keep the following points in mind when you create the THISHOST file:

- Each controller has both a device address and a unique Internet address. The lines in the file that describe the controllers can appear in any order.
- If the first line of the THISHOST file consists of only a semicolon or begins with a semicolon, PRIMOS TCP/IP uses the PRIMOS node name for HOST name.

Note

It is strongly recommended that you use the PRIMOS node name as the default name, provided that name meets the DoD naming standard. Using the PRIMOS name enables users connecting to the system via PRIMENET (remote login), NETLINK, LAN300 terminals, or PRIMOS TCP/IP to call the system using the same name. Using the default avoids the confusion of identifying one system by two names.

Subnetworking

PRIMOS TCP/IP supports the creation of subnetworks (subnets). Subnetworking lets you subdivide a large network into smaller units and assign each of these networks its own network number. Subnetworking can be in use on all, some, or none of the networks in a PRIMOS TCP/IP installation. The subnet to which your system belongs is identified in the THISHOST file.

Subnetworks have advantages when a large number of hosts are connected to the network. A subnetwork reduces both the load that a large number of nodes on a single network causes and the complexity of administering a single large network. However, a gateway sees a network that consists of multiple subnets as a single network.

Note

If your network currently does not have a subnetwork, it may not be necessary to implement one. Therefore, you can skip the next three sections.

How to Define a Subnetwork: To divide a network into subnets, you designate some bits of the 32-bit Internet host address as a subnetwork number. You can do this in several different ways. For example, on a Class B network, the network number occupies the most significant 16 bits of the host address. The host number occupies the least significant 16 bits. To divide a Class B network into four subnets, you can designate the two most significant bits of the host number as the subnet number. The following examples are host addresses of nodes that are on different subnets.

Example 1:

<i>Address</i>	<i>Description</i>
128.10.0.1	On subnet 0
128.10.64.1	On subnet 1 because 64 decimal is 01000000 binary
128.10.128.1	On subnet 2 because 128 decimal is 10000000 binary
128.10.192.1	On subnet 3 because 192 decimal is 11000000 binary

You do not have to specify the subnet field from the most significant bits of the host number and it is not always two bits wide.

Example 2:

<i>Address</i>	<i>Description</i>
128.10.0.0	On subnet 0
128.10.0.1	On subnet 1
128.10.0.2	On subnet 2
...	
...	
...	
128.10.0.7	On subnet 7

In Example 2, the subnet field is three bits wide to provide eight subnets. The subnet bits are in the least significant bits of the host number field. You are not restricted to the addressing schemes shown in the above examples. The subnet bits can appear anywhere in the address. You can designate only one bit as the subnet bit to provide only two subnets. Alternatively, you can use all the bits (except one) in the host field as subnet addresses. This leaves only one bit for the host number so that only two hosts can be connected to each subnet.

If your site supports subnets, you must specify to PRIMOS TCP/IP the number and location of the subnet bits with a subnet mask, described in the next section.

How to Specify a Subnet Mask: A subnet mask is a bit field in the THISHOST file containing a one (1) that corresponds to each bit of the network number field and the subnet field in the host address. The subnet mask of the host address 128.10.0.1 in Example 1 above is

11111111/11111111/11000000/00000000 (18 ones followed by 14 zeros)

It is easier to specify the subnet mask in hex:

```
FFFFC000
```

The mask for the host address 128.10.0.0 in Example 2 is

```
11111111111111110000000000000011 or FFFF0003
```

The following list shows some more examples:

<i>Mask</i>	<i>Description</i>
FF00000F	A Class A network with four subnet bits in the extreme right of the address.
FFC00003	A Class A network with four subnet bits, two at the left of the host number field and two at the right.
FFFFFF18	A Class C network with two subnet bits in the middle of the host number field.

How to Add the Subnet Mask to the THISHOST File: You specify subnets in the THISHOST file. The following example shows the format of a THISHOST file that specifies that SYS6 belongs to a subnet on a network connected to the controller with device address 53_g. The Internet address indicates a Class B network. The subnet field is four bits wide and occupies the most significant host number bits.

```
SYS6
device addr = 53 : internet address = 129.5.42.3 : subnet mask = FFFFF000 :
device addr = 54 : internet address = 10.2.3.4 :
```

The controller with physical device address 54_g has no subnet mask, which indicates that the network does not use subnets.

In the following example, the second network does use subnets. The Internet address indicates a class A network. If the subnet bits occupy the least significant eight bits, the THISHOST file contains the following entries:

```
SYS6
device addr = 53 : internet address = 129.5.42.3 : subnet mask = FFFFF000 :
device addr = 54 : internet address = 10.2.3.4 : subnet mask = FF0000FF :
```

For another example of a subnetwork, see the section entitled Sample Network 3 in Chapter 15.

The HOSTNAME_CONFIG File

PRIMOS TCP/IP supports the client portion of the Hostname Service protocol (described in RFC-953). The protocol enables a 50 Series system to use a TCP connection to retrieve a host

name or Internet address from the host table of a remote system that supports the server portion of the Hostname Service protocol. The information returned is in standard host table format. The Hostname Service protocol enables a Network Administrator to update the permanent host table without stopping and restarting TCP/IP.

When PRIMOS TCP/IP is started, the contents of the HOSTS.TXT configuration file are read into memory. The contents of this table are retained in memory even if the managing process is restarted. In addition to the permanent table, a **host cache** is created. The host cache consists of host entries retrieved from remote systems implementing the Hostname Service protocol. The HOSTNAME_CONFIG configuration file controls the characteristics of this cache.

Chapter 14, Activating PRIMOS TCP/IP, describes how to update the TCP/IP configuration files.

Note

At Release 2.0, PRIMOS TCP/IP does not support the Domain Name Service protocol. The Hostname Service uses port 101.

The Network Administrator creates a file that specifies

- How long cache entries can stay in memory (time to live)
- Instructions to maintain the cache even if PRIMOS TCP/IP is stopped and restarted
- Internet addresses of any Hostname Servers that are to be used

Format of the HOSTNAME_CONFIG File

The following example shows a HOSTNAME_CONFIG file that you can create with a text editor such as ED or EMACS:

```
CACHE AGE = 21
SAVE CACHE = FALSE
SERVER ADDRESS = 35.8.1.1
SERVER ADDRESS = 10.0.0.51
```

CACHE AGE, SAVE CACHE, and SERVER ADDRESS are keywords that introduce a line entry. The next sections describe the parameters entered after each keyword.

CACHE AGE: The parameter that follows the CACHE AGE keyword specifies the number of days for which information retrieved from a Hostname Server on a remote host remains valid. Entries from the local HOSTS.TXT file always are assumed to be valid and are not affected by this parameter. The default value is 14 days.

SAVE CACHE: The parameter that follows the SAVE CACHE keyword specifies whether the cache entries are saved when you issue either a STOP_TCP/IP or TCP/IP_INIT_HOSTS command. (The STOP_TCP/IP and TCP/IP_INIT_HOSTS commands are described in Chapter 14, Activating PRIMOS TCP/IP.)

If you omit either this line entry or omit the parameter, the default is TRUE. Saved cache entries are stored in the file TCP/IP*>HOSTNAME_CACHE.

SERVER ADDRESS: You can enter a maximum of four remote host (server) addresses. Each Internet address must be preceded by a SERVER ADDRESS keyword. If you enter more than one server address, remote hosts (Hostname Servers) are accessed in the order in which they appear in the file.

Note

You cannot enter the address of a 50 Series system running PRIMOS TCP/IP. PRIMOS TCP/IP does not support the server portion of the Hostname Service protocol.

If you omit the HOSTNAME_CONFIG file or you omit the server address, the PRIMOS TCP/IP Manager checks only the HOSTS.TXT file on your local 50 Series system for name/address translations. If your system is not connected to the Internet and no other system on your network provides a hostname service, you can omit the server address entries or the HOSTNAME_CONFIG file.

Summary

Keep the following points in mind when you create the HOSTNAME_CONFIG configuration file:

- Name the file HOSTNAME_CONFIG and save it in the TCP/IP* directory.
- You can enter all data in uppercase or lowercase.
- The entry lines can appear in any order. Server address entries that are listed first are read first.
- If your system is not connected to the Internet and no other system on your network supports the Hostname Service protocol, you do not have to create this file. Default values, described above, are substituted for each line entry.

Host Table Utility

After you create the HOSTS.TXT, THISHOST, and HOSTNAME CONFIG files, you must run a utility called the PRIMOS TCP/IP Host Table Utility (TCP/IPHTU) to prepare the information in these files for use by PRIMOS TCP/IP. The utility runfile, TCP/IPHTU.RUN, is located in the TCP/IP* directory.

Perform the following steps to run this utility:

1. Attach to the TCP/IP* directory.
2. Enter the following command to execute the TCP/IPHTU.RUN utility runfile:

R TCP/IPHTU [filename]

filename is the name of the PRIMOS TCP/IP configuration file. If you omit *filename*, TCP/IPHTU reads HOSTS.TXT. If you specify *filename*, the utility reads that file. TCP/IPHTU also reads the THISHOST file, rearranges some of the information in both files, converts entries in uppercase to lowercase, and writes the information to a set of files. The modified host information is written to a file that is always named HOSTS.

Gateway information is written to one binary file for each LHC present on the system. The format of the filenames is GATEWAYS_*nn*, where *nn* is the physical device address of the controller in octal. If files with these names already exist at the time that you run TCP/IPHTU, TCP/IPHTU deletes the old versions.

Note

TCP/IPHTU must read the THISHOST file to get information about the number of and device addresses of controllers and the Internet address associated with each. Therefore, you must create the THISHOST file before you run TCP/IPHTU.

Copying a UNIX HOSTS File

The format of a HOSTS file is identical to the format of a UNIX hosts file. If you copy a UNIX hosts file to a 50 Series system, copy the file after you run TCP/IPHTU. If you install a hosts file and then run TCP/IPHTU, your hosts file is overwritten.

TCP/IPHTU creates both a HOSTS file and a GATEWAYS_*nn* file. (If you have no gateways listed in the HOSTS.TXT file, the utility creates a null GATEWAYS_*nn* file.) TCP/IP requires both files in order to start.

Error Conditions

The following conditions cause TCP/IPHTU to fail and report an error:

- Either the HOSTS.TXT file or the THISHOST file or both do not exist or are formatted incorrectly.
- A name specified in the THISHOST file does not meet DoD naming criteria.
- The name is set to default to the PRIMOS name, but the PRIMOS name is not valid under DoD naming criteria.

Appendix D describes PRIMOS TCP/IP configuration and installation error messages.

Modifying the System Configuration File

Before PRIMOS TCP/IP is started, you must perform the following tasks to modify the system configuration file (CONFIG), which is stored in the CMDNC0 directory:

- Add the LHC directive to CONFIG
- Check the value of the NPUSR directive

LHC Directive

For each LHC dedicated to PRIMOS TCP/IP, you must add an LHC directive to the CONFIG file. The LHC directive links a physical device address with an LHC logical controller number. The directive takes effect when the CONFIG file is processed at cold start.

The LHC directive has the following format:

LHC controller-number device-address

controller-number

Specifies the LHC's logical controller number. You must select a *controller-number* that is in the range 0 through 7, inclusive. Each LHC dedicated to PRIMOS TCP/IP, PRIMENET, or NTS must be assigned a unique logical number.

device-address

Specifies the LHC's physical device address in octal. Physical device addresses are assigned to LHCs from a valid range of addresses and are a maximum of two octal digits. *device-address* is the number that you entered in the THISHOST file.

NPUSR Directive

The NPUSR directive sets the number of phantom users. The NPUSR directive has the following format:

NPUSR total_number

total_number is the total number (in octal) of phantom users for the system to be configured. PRIMOS TCP/IP uses phantoms from the system pool of phantoms. Therefore, you must increase the number of phantoms for PRIMOS TCP/IP. Use the following formula to determine the number of phantoms that you must add for PRIMOS TCP/IP:

$$number \Rightarrow (C + 1) + D + M + R + W + N$$

where

- C** = The number of LHCs dedicated to PRIMOS TCP/IP. Each controller has one server phantom (FTP_SERVER nn).
- 1** = One phantom for the MAIL process (SMTP_SENDER0). Each system has one MAIL process phantom.
- D** = One for the MAILER_DAEMON that transfers mail to users.

- M = One for the MAIL process (SMTP_SERVERnn) that receives messages from the network.
- R = Number of remote sessions from the network.
- W = One for the TCP/IP process (TCP/IP_MANAGER) that aids recovery.
- N = One for the LAN300 Network Management server (NM_SERVER).

To specify the number of remote sessions (R in the formula), estimate the maximum number of actual (not potential) remote FTP users.

One Network Management server works on behalf of both NTS and PRIMOS TCP/IP. Therefore, add a phantom for NM_SERVER (N in the formula) for either PRIMOS TCP/IP or NTS but not both. Only one NM_SERVER runs on a host.

Configuring Network Terminal Service for PRIMOS TCP/IP Users

The TELNET program requires that the Network Terminal Service be both installed on the 50 Series host and configured for PRIMOS TCP/IP. The instructions contained in the next sections describe NTS parameters and screens that apply directly to PRIMOS TCP/IP. These instructions assume that NTS has not been installed.

If NTS has been installed and configured for NTS terminals, see the section entitled Configuring PRIMOS TCP/IP When NTS Is Installed, later in this chapter. For complete information on installing and configuring NTS, see the *NTS Planning and Configuration Guide*.

After NTS is installed, invoke the NTS configuration utility with the CONFIG_NTS -CREATE command and perform the following steps, which are described in the next sections:

1. Display the CONFIG_NTS main menu.
2. Enter information on the LAN Configuration screen.
3. Enter the name of the 50 Series host on the Configure Host on the LAN screen and save the configuration file.

CONFIG_NTS Main Menu

To display the CONFIG_NTS main menu, enter the following command and optionally, the name of the NTS configuration file:

```
OK, CONFIG_NTS -CREATE
```

If you do not enter the pathname of the configuration file, NTS prompts you for it. You can accept the default pathname (NTS*>NTS.CONFIG) by pressing without making an entry. CONFIG_NTS then prompts you for one or two lines of descriptive text.

Refer to the *NTS Planning and Configuration Guide* for a complete description of the NTS command format and options.

NTS displays the NTS main menu as shown in Figure 12-6.

```
Create configuration file: <NTS*>NTS.CONFIG

Sample configuration file for PRIMOS TCP/IP.

  1. Configure LAN                H. Help
  2. Configure Host by LAN       S. Save current configuration
  3. Configure Host by Name      Q. Quit configuration session
  4. Configure LTS by LAN
  5. Configure LTS by Name
  6. Display, list or spool configuration
  7. Change configuration title

Enter selection: 1
```

Figure 12-6
CONFIG_NTS Main Menu

Enter selection 1 and press to display the LAN Configuration screen. See Figure 12-7.

LAN Configuration Screen

The LAN Configuration screen, illustrated in Figure 12-7, lists LAN300 configuration functions.

```
Configuration file: NTS*>NTS.CONFIG

Sample configuration file for PRIMOS TCP/IP.

No LANS are configured.

Enter:
    A to add a LAN
    F to finish (return to main menu)
    S to save current configuration
    Q to quit configuration session

Enter selection: A
```

Figure 12-7
LAN Configuration Screen

Perform the following steps:

1. Enter A on the LAN Configuration screen. CONFIG_NTS prompts you for the name of the LAN, suggesting a default name in the form LAN300 nn , where nn is a number from 1 to 32, inclusive.
2. Enter either the name of the LAN300 or press without making an entry to accept the default name. For consistency and to avoid confusion, use the same name that is used in the PRIMENET configuration, if you have one.
3. When CONFIG_NTS prompts you for two lines of descriptive text, you can enter a maximum of two 80-character lines of descriptive text about this LAN.
4. When CONFIG_NTS displays the query, Allow unconfigured nodes on this LAN?, enter YES.

An unconfigured node is one that is attached to the LAN300, but not included in the configuration file. Remote systems running TELNET look like unconfigured nodes to the 50 Series host. NTS software on the 50 Series host does not check whether the address of an unconfigured LTS is within the range of addresses assigned to Prime LTSs.

Caution

Do not configure an LTS on the network to emulate a remote system running TELNET. Allowing unconfigured nodes on the network provides this function. All LTSs on the network are expected to respond to LAN300 network management commands. A remote system that is not an LTS will not respond; including it in the configuration file might disrupt the operation of other valid NTS nodes.

5. CONFIG_NTS displays this prompt:

Configure network management functions for this LAN?

Enter NO if the 50 Series host is running only PRIMOS TCP/IP.

The network management section of the LAN300 configuration indicates which hosts on the LAN300 respond to LTS requests for downline loads and upline dumps of software. It also indicates which hosts collect event and error reports sent by the LTSs on the network. These reports document downline loads, upline dumps, or any kind of error condition.

PRIMOS TCP/IP cannot provide these services to LTSs. If a 50 Series host is running only PRIMOS TCP/IP, do not configure the host as either a downline load, upline dump, or event reporting host. If the 50 Series host has one or more LHCs dedicated to NTS or PRIMENET, you can configure the 50 Series system for LAN300 network management. Return to the main menu and enter selection 2 (Configure Host by LAN) to display the LAN Selection screen. When you have only one LAN300 configured, the Configure Host by LAN screen is displayed, as described in the next section.

Configuring Hosts on a LAN

The Configure Host on LAN screen, shown in Figure 12-8, enables you to associate a LAN with a host name and save the configuration file.


```
Configure host on LAN: LAN300-1

Default name of LAN to which Prime and non-Prime system connected.

No hosts are configured on this LAN.

Enter:
  A to add a host to the LAN      F to finish (return to prev menu)
                                  S to save the current configuration
                                  Q to quit the configuration session

Enter selection: A
```

Figure 12-8
Configure Host on LAN Screen

When the Configure Host on LAN screen appears, perform the following steps:

1. Enter selection A to add a host to the LAN300.
2. CONFIG_NTS prompts you for the name of the first LAN300 to which the host is connected. CONFIG_NTS then prompts you for the host's name.
3. Enter the name that is set at cold start (with the SYSNAM configuration directive). In most cases, this is the name that you entered in the THISHOST file.
4. Next, CONFIG_NTS prompts you for a list of LHC names in the host connected to the LAN300 that you just named. LHC names must correspond to the logical numbers entered in the configuration file with the LHC directive. For example, LHC0 must correspond to LHC00.
5. Return to the CONFIG_NTS main menu and save the configuration file.

Configuring PRIMOS TCP/IP When NTS Is Installed

This section describes how to configure PRIMOS TCP/IP on a 50 Series system on which NTS and the LAN300 Network Management facility have been installed and configured. Perform the following steps:

1. Install PRIMOS TCP/IP software as described in Chapter 11, Installing PRIMOS TCP/IP.
2. Create the configuration file (HOSTS.TXT) according to instructions described earlier in this chapter.
3. Create the THISHOST file to identify the local host according to instructions provided earlier in this chapter.
4. Create the HOSTNAME_CONFIG file for the Hostname Service according to instructions provided earlier in this chapter.
5. Run the Host Table Utility (TCP/IPHTU) according to instructions provided earlier in this chapter.
6. Modify the system configuration file as described in this chapter and Appendix F, NTS-related Configuration Directives.
7. Invoke the NTS configuration utility with the CONFIG_NTS command. Edit the NTS configuration file if necessary. Be certain to permit unconfigured nodes when the LAN Configuration screen (Figure 12-7) appears.

Chapter 14, Activating PRIMOS TCP/IP, describes how to start NTS.

Configuring DSM for PRIMOS TCP/IP Unsolicited Messages

Distributed Systems Management (DSM) supports event logging from networked systems. The DSM Unsolicited Message Handler (UMH) facility enables you to route event messages to a device or to a log file on any system that is a PRIMENET node. Use the utility CONFIG_UM to determine the route of all messages and their destination. For more information on the CONFIG_UM utility, refer to the *DSM User's Guide*.

Note

Configuring DSM for PRIMOS TCP/IP is optional. PRIMOS TCP/IP starts and runs without invoking CONFIG_UM.

Use the following command to define a new selection:

CONFIG_UM *selection option*

where *selection* is a selection name with a maximum of 32 characters and *option* describes the operation that you want to perform. To define a new selection for PRIMOS TCP/IP, use the `-SELECT` option.

After you enter the `CONFIG_UM` command with the `-SELECT` option, you are prompted to enter information for each of the following items:

- Products
- Severities
- Destinations

To enter an item, type it after the appropriate prompt, and press . Enter TCP/IP for the product name. It is recommended that you enter `-ANY` to include all severity levels.

Example of Message Routing

In the following dialog, the Network Administrator routes messages of all severity levels from PRIMOS TCP/IP on SYS7 to the administrator's terminal and a private log `TCP/IP*>LOG.FILES>TCP.LOG`. The Network Administrator works under the user name `SYSTEM`.

```
OK, CONFIG_UM EXAMNET -SELECT -ON SYS7
Product name: TCP/IP
Product name: 
Severity: -ANY
Severity: 
Destination: DISPLAY -USER SYSTEM -ON SYS7
Destination: LOGGER -PRIVATE_LOG TCP/IP*>LOG.FILES>TCP.LOG -ON SYS7
Destination: 
Do you wish to edit this selection ? NO
Configuring EXAMNET on SYS7
Completed OK
OK,
```

If PRIMOS TCP/IP is installed after DSM is installed, you must grant DSM access to the `LOG.FILES>TCP.LOG` file in the `TCP/IP*` directory. Use the `EDIT_ACCESS` command (abbreviated `EDAC`) to give the DSM logging server (DSMASR) `PDALURWX` access to the `TCP/IP*>LOG.FILES>TCP.LOG` file, as in the following example:

```
OK, EDAC <partition_name>TCP/IP*>LOG.FILES>TCP.LOG DSMASR:PDALURWX
```


Configuring PRIMOS TCP/IP Mail

This chapter describes how a Network Administrator or operator configures PRIMOS TCP/IP MAIL. MAIL requires that the HOSTS.TXT and the THISHOST file be created and formatted with the TCP/IPHTU utility before you start MAIL. Therefore, read Chapter 12, Configuring TCP/IP, before you attempt to configure MAIL.

The major task of configuring MAIL is creating the MAIL configuration file, which contains both required and optional parameters. The parameters and values that you enter in this file depend on topology of the network or networks to which your system is connected. Some parameters establish the time intervals between which MAIL attempts to send mail and to poll other systems to establish routes. Other parameters identify systems that can serve as relay nodes (intermediate systems).

In addition to configuring MAIL, this chapter describes how you create mailboxes for users. You must do this only if you do not allow users to create password directories.

The chapter begins with a description of how MAIL works as a store-and-forward system.

MAIL Operations

MAIL is a store-and-forward system that can perform the following tasks:

- Store mail submitted by local users temporarily before sending it over the network
- Store mail received from an adjacent system and forward it to a final destination
- Act as a relay for mail sent between two networks

The destination of mail can be a user on the local system or a user on a remote system connected to

- A LAN300
- The Internet
- A RINGNET (by means of PRIMENET)
- A packet switching data network (PSDN) (by means of PRIMENET)
- Any other PRIMENET media (for example, a leased line)

Figure 13-1 shows a sample network over which MAIL can route messages.

Note

A 50 Series system connected to a RINGNET or a PSDN (but not connected to a LAN300) can send and receive MAIL messages only if it is running the MAIL software and either PRIMENET or PRIMENET X.25.

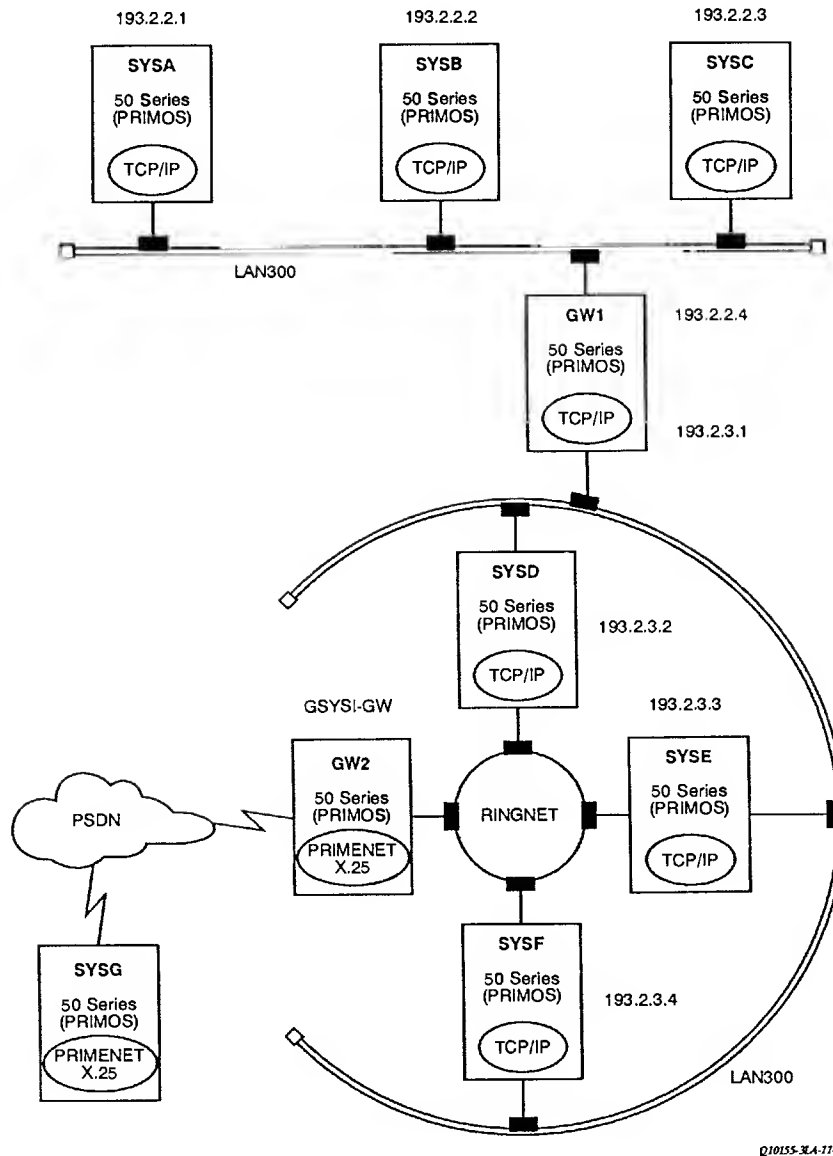


Figure 13-1
Sample MAIL Network

Local Operation

A user on a 50 Series system creates a mail message to be delivered to either a user on the local system or a user on a remote system. The MAIL user interface opens a connection to the MAILER_DAEMON and transmits the message to it. The MAILER_DAEMON checks whether the recipient is on the local system.

If the recipient is a local user, the MAILER_DAEMON attempts to put the message in the user's mailbox. If this operation fails, the sender is notified immediately and the message is placed in a DEAD.LETTER file.

MAIL can also route messages to a user on a remote system.

Routing to a User on a Remote System

If the destination of the message is a user on a remote system, the MAILER_DAEMON constructs a path to the destination system using its internal routing table. The routing table comprises information from the following sources:

- The HOSTS.TXT file
- The MAIL configuration file
- The PRIMENET configuration file (if present)
- Return paths from previously received mail

The MAILER_DAEMON constructs a route that may include several intermediate systems to reach the final destination system. These intermediate systems can be systems to which incoming mail was sent or systems that are listed in the MAIL configuration file.

If the first system on the route is a PRIMENET node, MAIL sends the message via PRIMENET to the MAILER_DAEMON on that system. If the destination system or the first intermediate system is on a LAN300 and runs PRIMOS TCP/IP, the message is placed in an outgoing queue. The SMTP_SENDER0 process picks up the message and attempts to send the message to the destination system or an intermediate system. The SMTP_SENDER0 process must have either the IP address of that system (from the HOSTS.TXT file) or it must be able to use the Hostname Service to obtain the address from the Hostname Service of another system.

If the attempt to deliver the message fails, MAIL tries to send the message again at intervals that you specify in the configuration file. (The default retry value is 20 minutes.) You also specify in the configuration file how long MAIL attempts to deliver the message before informing the sender that the message cannot be delivered.

When a destination system can be reached by way of either PRIMOS TCP/IP or PRIMENET (as shown in Figure 13-1), MAIL first selects PRIMENET as a route.

MAIL stores routing information in files named ROUTE.CACHE and HOST.CACHE in the MAILER*>DB directory. MAIL makes calls to other PRIMENET systems to create the HOST.CACHE file and creates the ROUTE.CACHE file when users send and receive messages.

Receiving and Forwarding Messages

Upon receiving mail, a host reads the destination address of the recipient. If the recipient is a local user, MAIL puts the message in its incoming queue. The MAILER_DAEMON puts the message in the recipient's mailbox from which the user retrieves it by issuing a MAIL command.

If the address of the recipient is an address of another system that can be reached directly, MAIL attempts to send the message to the recipient's system. If the destination host can be reached only by way of an intermediate system or a gateway, MAIL sends the message to that system.

Required MAIL Configuration Parameters

The MAILER*>DB>CONFIGURATION.TEMPLATE file, which is created when MAIL is installed, is the template of the MAIL configuration file. Modify the entries in this file with a standard Prime editor, such as ED or EMACS. Entries can be in uppercase or lowercase.

The default MAIL configuration file has the following entries:

- domain *.none.COM*
- relay-name *hostname.none.COM*
- postmaster *user-ID*

The next sections describe these parameters.

Domain Parameter

The domain parameter in the configuration file consists of the following entry:

domain *.none.COM*

domain is the keyword that precedes the Domain entries. This parameter identifies the domain to which your system belongs.

In place of *none*, supply the **site name** for your organization. Site name is usually the name of your company or organization. In place of *.COM*, supply the appropriate top-level domain name. For example, if your site is a company in the United States named ACME, *.ACME.COM* is the appropriate entry. If your organization is a university, the suffix *.EDU* may be the appropriate entry. The domain entries must also appear as part of the first name in the HOSTS.TXT file.

If you are connected to the Internet, you must obtain the appropriate top-level domain name from the NIC. If you are not connected to the Internet, you can accept the default (*.COM*) or create your own domain name. You can configure your system to be in any domain that you create. A domain name can be a maximum of 255 characters. Each element (for example, site name) can be a maximum of 63 characters.

Note

Systems to which you want to send mail must be listed in the HOSTS.TXT file with their domain names.

The following list gives the meanings of some of the domains to which your system can belong:

<i>Domain</i>	<i>Description</i>
.ARPA	The DARPA Internet
.AU	Australia
.OZ.AU	Australian academic community
.CA	Canada
.COM	Commercial companies in the United States
.DE	West Germany
.dpb.DE	Deutsches Forschungnetz
.EDU	Educational institutions in the United States
.FAX	The international direct-dial facsimile system
.GOV	United States government agencies
.IL	Israel
.ac.IL	The academic community in Israel
.JP	Japan
.KR	South Korea
.ac.KR	The academic community in South Korea
.Mail	The postal mail system
.MIL	U.S. Department of Defense
.bitnet	The BITNET
.cs.net	The computer science network maintained by BBN
.NSF.net	The network of the United States National Science Foundation
.uu.net	A UNIX network
.NO	Norway
.NZ	New Zealand
.ac.NZ	The academic community in New Zealand
.ORG	Other organizations, not elsewhere classified
.PDN	Companies and services reachable via public data networks
.SE	Sweden

.Telex	The international TELEX system
.UUCP	The UNIX-to-UNIX copy network
.UK	The United Kingdom
.ac.UK	The UK academic community (JANET)

Sub-site Name: If the name of your 50 Series system is not unique, you may need to specify a sub-site name. A sub-site name distinguishes a host in one location from an identically named host in another location. If a company named ACME has a host named SYS1 in the U.S.A. and another host named SYS1 in the United Kingdom (U.K.), sub-site names are needed to distinguish the two hosts.

The following entry is for a host with a unique name (SYS1) at site ACME in the U.S.:

```
domain .ACME.COM
```

In the above example, .ACME replaces the *none* parameter.

The following entry is for a host with a non-unique name (SYS1) at site ACME in the United Kingdom:

```
domain .ACME.UK
```

In the above example, .ACME.UK replaces the *none* and .COM entries.

Relay Node Parameter

Use the relay node parameter to specify a fall-back system ("smartest neighbor") for sending mail when the local node is not aware of an appropriate route. If the fall-back system knows the appropriate route, it routes the message to the target host.

The relay node parameter has the following line entry:

```
relay-name hostname.none.COM
```

where **relay-name** is the keyword that precedes the parameter.

To configure this parameter,

- Replace *hostname* with the host name for a fall-back system listed in the HOSTS.TXT file. The name must meet DoD naming conventions.
- Replace *none* with the site name, and if applicable, the sub-site name specified for the Domain parameter.
- Replace .COM with another top-level domain name if appropriate.

Specify the local host if that system stores all the routing information.

Postmaster Parameter

The Internet protocol standard requires a postmaster parameter for sending mail to the Internet. If you have problems sending mail to a remote site, you can send mail to the address postmaster at that site and that person can direct your message to its destination. Every mail system should designate a user who serves as postmaster.

The parameter has the following format:

postmaster *user-ID*

postmaster is the keyword and *user-ID* is the user ID of the person who is responsible for administering and maintaining the electronic mail service. For example, if the user ID of the postmaster of SYS1 is NAMLIG, the postmaster entry in the configuration file of SYS1 is

```
postmaster NAMLIG
```

If the postmaster of SYS1 is NAMLIG on SYS6, the postmaster entry in the SYS1 configuration file is

```
postmaster NAMLIG@sys 6
```

You can specify the address POSTMASTER if that user ID exists as in the following example:

```
postmaster postmaster@sys 6
```

Name this file CONFIGURATION and save it in the MAILER*>DB directory.

Note

MAIL may not work correctly without a valid configuration file. If users attempt to send MAIL without a configuration file, MAIL creates invalid CACHE files that prevent routing messages correctly. To remedy the problem, stop the MAILER_DAEMON, delete any CACHE files in the MAILER*>DB directory, create a configuration file, and then restart the MAILER_DAEMON.

Optional Time and Transmission Parameters

The parameters described in this section pertain to default values for parameters that establish the time intervals between which MAIL attempts to send mail and to poll other systems to establish routes. MAIL polls other PRIMENET nodes to determine whether the nodes are running MAIL. MAIL uses the information it gathers to construct a path to a remote system or an alternate path if a route is blocked. Note that some time and transmission parameters apply only to PRIMENET nodes while others apply to both PRIMENET and PRIMOS TCP/IP nodes.

Change any of the default settings if required. Add these parameters to the MAILER*>DB>CONFIGURATION file.

Enter parameters in any order. Blank lines between entries are ignored. Lines beginning with an asterisk (*) are treated as comments and are also ignored.

The parameters described in this section

- Supply information to the message header that describe when messages are sent and received
- Determine the intervals for calls and message relays to other nodes

Each parameter entry has the following format in the configuration file:

parameter_name value

Specify the values for intervals in units of days, hours, minutes, or seconds, as illustrated here. (You can abbreviate time units: m for minutes, s for seconds, and so on.)

```
parameter_name 120 min
parameter_name 65 seconds
parameter_name 3 hours
parameter_name 4 days
```

Table 13-1 describes the parameters and their meanings. The default values for each parameter are shown in Table 13-2.

Table 13-1
Time and Transmission Parameters

<i>Parameter</i>	<i>Description</i>
Transport-scan-interval	Controls the maximum interval between scans of the incoming message queue of gateways (relays) by the MAILER_DAEMON. You configure gateways with the add-transport parameter described in the section entitled Node and Routing Parameters. The transport-scan-parameter determines how often mail is collected from the SMTP incoming mail queue.
Local-area-interval	Determines the interval between attempts to send mail to another PRIMENET node. A short interval is typical for this parameter because costs are not incurred for each connection. A short interval results in frequent connections and little delay in forwarding messages.
Local-area-retry	Determines the interval before a failed attempt to send mail is retried.

Table 13-1
Time and Transmission Parameters – Continued

<i>Parameter</i>	<i>Description</i>
Local-area-search	Controls the interval between polling calls to other local PRIMENET nodes. MAIL polls PRIMENET nodes independently of its sending or receiving messages. MAIL checks whether other PRIMENET nodes have a MAIL program. MAIL uses the information gathered from polling to configure a route when it sends mail messages to that node. The value of this parameter should be different from other time-based events. For example, an interval of 24 hours might call a system every day exactly when it is shut down. A better choice would be 23 or 25 hours.
Log-file-reset	Controls how long the log files in the MAILER*>LOG directory are retained before the MAILER_DAEMON deletes them. If you specify a value of 0 seconds (or anything less than the value you entered for the log-file-reset parameter), only the current and previous log files are retained since the MAIL program was last restarted. (All log files are deleted when the system is restarted.)
Message-timeout	Determines the interval for attempting to forward a message before returning it to the sender. The interval is always 3 days (the default) when the destination of a message is an SMTP node. The interval for an X.25 node should be at least three times larger than the intervals of the local-area-retry parameter and wide-area-retry parameter.
Wide-area-interval	Determines the interval between calls to an X.25 node for which you pay usage tariffs. A longer interval than that for the Local-area-interval parameter is typically used because it results in fewer connections (but greater delay in forwarding messages).
Wide-area-retry	Determines the interval before a failed attempt to send mail to a node is retried.

Table 13-1
Time and Transmission Parameters – Continued

<i>Parameter</i>	<i>Description</i>
Wide-area-search	Determines the interval between polling calls to other X.25 nodes. MAIL polls PRIMENET nodes independently of its sending or receiving messages. MAIL checks whether the X.25 node listed in the PRIMENET configuration file has a MAIL program. MAIL uses the information gathered from polling to configure a route when it sends mail messages to that node. The interval that you choose should be different from other time-based events. For example, a call to another node every 24 hours might occur just when that system is shut down. A better choice would be 23 or 25 hours. Do not set a value of less than one week for the wide-area-search parameter without being aware of the result. A setting of 5 minutes, for example, calls other systems in your configuration every five minutes until they respond. The cost for all of these calls would be very expensive. The capability of setting a short interval for this parameter is provided for customers who have their own packet switching networks.

Table 13-2 describes the default values for the time and transmission parameters described above.

Table 13-2
Default Values for Time and Transmission Parameters

<i>Parameter</i>	<i>Default</i>	<i>Minimum</i>	<i>Maximum</i>
Transport-scan-interval	5 minutes	30 seconds	1 day
Local-area-interval	30 seconds	1 second	10 minutes
Local-area-retry	20 minutes	1 minute	1 hour
Local-area-search	9 hours	1 minute	5 days
Log-file-reset	1 day	1 hour	10 days
Log-file-retention	7 days	0 seconds	1000 days
Message-timeout	3 days	1 hour	5 days
Wide-area-interval	15 minutes	1 second	2 hours
Wide-area-retry	1 hour	1 minute	1 day
Wide-area-search	9 days	1 minute	1000 days

Optional Node and Routing Parameters

The following optional parameters establish routing and addresses for MAIL messages:

Block-primenet

The `block-primenet` parameter blocks all calls to a PRIMENET node that is listed in the PRIMENET configuration file but does not have MAIL installed.

The format of the line entry for the `block-primenet` parameter is

```
block-primenet nodename
```

where `block-primenet` is a keyword and *nodename* is the name of the host to which calls are to be blocked.

Block-tcp/ip

The `block-tcp/ip` parameter blocks all calls to either a domain or a specific node on a LAN300 that is listed in the HOSTS.TXT file but does not have MAIL installed.

The format of the line entry for `block-tcp/ip` parameter is

```
block-tcp/ip domain_name
```

where `block-tcp/ip` is a keyword and *domain_name* is either a domain name or the official or alias name of a host (as listed in the HOSTS.TXT file) to which calls are to be blocked.

For example, the following entry blocks all calls to all systems in the .acme.COM domain:

```
block-tcp/ip .acme.COM
```

The following entry blocks calls only to SYSA:

```
block-tcp/ip sysa
```

Add-x25-address

The `add-x25-address` parameter specifies a node that is not listed in a your system's PRIMENET configuration file. Use this parameter only if your system will be sending mail to a system across a PSDN that runs MAIL. A PSDN line must be connected directly to your node.

The format of the line entry for the `add-x25-address` parameter is

```
add-x25-address nodename domain PRID
```

where `add-x25-address` is a keyword.

nodename is the name of the node that you want to add to the configuration file. *domain* is the domain name of the destination. The domain name is usually a site name followed by a suffix (for example, .Prime.COM).

PRID is a protocol identification number.

Add-route

The add-route parameter explicitly controls message routing. This parameter adds a route to the information that MAIL collects when it polls other systems. The format of the line entry for the parameter is

add-route *relay-node target-node*

where **add-route** is a keyword, *relay-node* is the system to which messages are to be routed, and *target-node* is the ultimate destination of the messages.

In the following example, PRIMOS TCP/IP MAIL routes messages addressed to M.I.T. to node *sys1.abc.COM*:

```
add-route sys1.abc.COM .mit.EDU
```

Note that the domain address (*.mit.EDU*) usually begins with a period (.) if it is not an entire host name. If you omit the period, MAIL supplies one.

You can add as many entries as you need. Each entry must be preceded by the **add-route** keyword.

Add-transport

The add-transport parameter configures a gateway for mail addressed to locations within a domain. A transport is defined here as a service program that receives files from the MAILER_DAEMON and sends them to a remote system over a medium. The service program also receives files from remote systems and passes them to the MAILER_DAEMON. This parameter ensures that mail directed to a specified domain is sent over a LAN300 via SMTP rather than over PRIMENET.

The line entry has the following format:

add-transport *domain service*

where **add-transport** is a keyword that introduces the parameters.

domain is the domain address, which normally begins with a period (.) if it is not an entire host name. If you omit the period, MAIL supplies one. *service* specifies the transport service supported. Currently, only SMTP_LAN300 is supported.

The following example shows an add-transport entry:

```
add-transport .abc.COM SMTP_LAN300
```

Hostname: The Hostname parameter specifies explicitly the entire host name of your 50 Series system. The parameter bypasses the designation that MAIL generates either from the SYSNAM parameter in the PRIMOS config file or from the entry in the THISHOST file. Use this parameter either when the host system has an assigned DARPA Internet name or when you want to replace a PRIMENET name. The name that you specify must be the name specified in the THISHOST file.

It is recommended that you use the Hostname parameter only on systems that run PRIMENET, not TCP/IP.

In the following example, CHI-A replaces the PRIMENET name CHI.A:

```
hostname CHI-A
```

Note

A node name can be longer than the six characters allowed in a PRIMENET name. For example, a PRIMENET node named CHI.A can be changed to CHICAGO-A.ABC.COM.

Sample MAIL Configuration Files

The following files are examples of the HOSTS.TXT, THISHOST, and MAIL configuration files for the systems shown in Figure 13-1. When a configuration file omits a parameter, default values are inserted in its place.

The configuration files share the following assumptions:

- All systems are in the same domain (.acme.COM).
- SYSA, SYSB, and SYSC are in one building (building 1); SYSD, SYSE, SYSF are in another (building 2); SYSG is in a third building (building 3).
- System GW1 is being used as a mail relay, not an IP gateway (which would allow systems in building 1 to talk directly to systems in building 2).

SMTP delivers mail over a LAN300 in building 1, while mail in building 2 travels over PRIMENET. SYSD serves as the relay between GW2 and GW1.

In addition to the MAIL configuration files, the HOSTS.TXT files for each system connected to the LAN300 must contain the addresses of the 50 Series systems connected to the LAN300. The next section describes these files.

HOSTS.TXT File for SYSA, SYSB, SYSC

The HOSTS.TXT files of systems SYSA, SYSB, and SYSC must contain addresses for SYSA, SYSB, SYSC and GW1. The complete domain name of each system must be listed first as the official host name.

```
host : 193.2.2.1 : sysa.building1.acme.com, sysa : : : :
host : 193.2.2.2 : sysb.building1.acme.com, sysb : : : :
host : 193.2.2.3 : sysc.building1.acme.com, sysc : : : :
host : 193.2.2.4 : gw1.building1.acme.com, gw1 : : : :
```

HOSTS.TXT File for GW1

The HOSTS.TXT file of system GW1 must contain addresses for SYSA, SYSB, SYSC, SYSD, SYSE, and SYSF. The complete domain name of each system must be listed first as the official host name.

```
host : 193.2.2.1 : sysa.building1.acme.com, sysa : : : :
host : 193.2.2.2 : sysb.building1.acme.com, sysb : : : :
host : 193.2.2.3 : sysc.building1.acme.com, sysc : : : :
host : 193.2.2.4, 193.2.3.1 : gw1.building1.acme.com, gw1 : : : :
host : 193.2.3.2 : sysd.building2.acme.com, sysd : : : :
host : 193.2.3.3 : syse.building2.acme.com, syse : : : :
host : 193.2.3.4 : sysf.building2.acme.com, sysf : : : :
```

HOSTS.TXT File for SYSD, SYSE, SYSF

The HOSTS.TXT files for systems SYSD, SYSE, and SYSF must contain addresses for SYSD, SYSE, SYSF and GW1. The complete domain name of each system must be listed first as the official host name.

```
host : 193.2.3.1 : gw1.building1.acme.com, gw1 : : : :
host : 193.2.3.2 : sysd.building2.acme.com, sysd : : : :
host : 193.2.3.3 : syse.building2.acme.com, syse : : : :
host : 193.2.3.4 : sysf.building2.acme.com, sysf : : : :
```

THISHOST File for SYSA

The THISHOST file for SYSA has the following entries:

```
sysa
device address = 56 : internet address = 193.2.2.1 :
```

THISHOST File for SYSB

The THISHOST file for SYSB has the following entries:

```
sysb
device address = 32 : internet address = 193.2.2.2 :
```

THISHOST File for SYSC

The THISHOST file for SYSC has the following entries:

```
sysc
device address = 56 : internet address = 193.2.2.3 :
```

THISHOST File for GW1

The THISHOST file for GW1 has the following entries:

```
gw1
device address = 56 : internet address = 193.2.2.4 :
device address = 32 : internet address = 193.2.3.1 :
```

THISHOST File for SYSD

The THISHOST file for SYSD has the following entries:

```
sysd
device address = 32 : internet address = 193.2.3.2 :
```

THISHOST File for SYSE

The THISHOST file for SYSE has the following entries:

```
syse
device address = 32 : internet address = 193.2.3.3 :
```

THISHOST File for SYSF

The THISHOST file for SYSF has the following entries:

```
sysf
device address = 56 : internet address = 193.2.3.4 :
```

MAIL Configuration File for SYSA

SYSA, in Acme building 1, has the following MAIL configuration file:

```
domain .building1.acme.com
relay-name gw1
postmaster dave
```

If SYSA does not know the route to a destination, it routes the message to GW1.

MAIL Configuration File for SYSB

SYSB, in Acme building 1, has the following MAIL configuration file:

```
domain .building1.acme.com
relay-name gw1
postmaster tony
```

If SYSB does not know the route to a destination, it routes the message to GW1.

MAIL Configuration File for SYSC

SYSC, in Acme building 1, has the following MAIL configuration file:

```
domain .building1.acme.com
relay-name gw1
postmaster cora
```

If SYSC does not know the route to a destination, it routes the message to GW1.

MAIL Configuration File for GW1

System GW1, in Acme building 1, has the following MAIL configuration file:

```
domain .building1.acme.com
relay-name sysd
postmaster dave
```

If GW1 does not know the route to a destination, it routes the message to SYSD.

MAIL Configuration File for SYSD

SYSD, in Acme building 2, has the following MAIL configuration file:

```
domain .building2.acme.com
relay-name sysd
postmaster carol
add-route gw1 .building1.acme.com
add-route gw1 .building3.acme.com
```

Note that entries for the relay node parameter in the configuration files of the other systems on the ring designate SYSD as the "smartest neighbor". SYSE and SYSF route messages to SYSD, which in turn, routes messages to the two gateway systems. SYSD designates itself as the relay node parameter because it can route messages to gateways.

MAIL Configuration File for SYSE

SYSE, in Acme building 2 and in the Eastern time zone, has the following MAIL configuration file:

```
domain .building2.acme.com
relay-name sysd
postmaster mike
```

If SYSE does not know the route to a destination, it routes the message to SYSD.

MAIL Configuration File for SYSF

SYSF, in Acme building 2, has the following MAIL configuration file:

```
domain .building2.acme.com
relay-name sysd
postmaster bernie
```

If SYSF does not know the route to a destination, it routes the message to SYSD.

MAIL Configuration File for GW2

GW2, in Acme Building 2 and in the Eastern time zone, has the following MAIL configuration file:

```
domain .building2.acme.com
relay-name sysd
postmaster john
add-x25-address sysg .building3.acme.com 1
```

MAIL Configuration File for SYSG

SYSG, in Acme building 3, has the following MAIL configuration file:

```
hostname sysg
domain .building3.acme.COM
pdn-relay-name sysg.building3.acme.COM
postmaster sysg_admin
add-x25-address gw2
add-route gw2 .acme.COM
```

Modifying the MAIL Configuration File

If you want to change required or optional parameters in the MAIL configuration file after MAIL has started, perform the following steps:

1. Stop the MAILER_DAEMON
2. Edit the configuration file
3. Delete MAILER*>DB>CACHE files (optional)
4. Start the MAILER_DAEMON

If you change parameters that affect routing, you must delete the ROUTE.CACHE and HOST.CACHE files in the MAILER*>DB directory before you restart the MAILER_DAEMON. Parameters that affect routing include the domain parameter and the relay node parameter.

MAIL stores routing information in the CACHE files. The routing information is derived from entries in the configuration file. Changing the value of a parameter in the MAIL configuration file can make a route obsolete or incorrect. If obsolete CACHE files exist, MAIL still accesses them to determine a route to a destination. After you delete a CACHE file, MAIL creates a new file based, in part, on current entries in the configuration file.

Creating Mailboxes

User mailboxes are subdirectories under the MAILER*>MBOX directory. Users can initialize their mailboxes only if password directories are allowed. The PASSWORD_DIRS command with the -ON option in the PRIMOS.COMI file specifies that the creation of password directories is allowed. This is the default if the PASSWORD_DIRS command is not specified at cold start. The PASSWORD_DIRS command with the -OFF option prevents users from initializing their mailboxes.

The following messages indicate that your system does not permit the creation of password directories:

```
OK, MAIL
[MAIL Rev. 22.0 Copyright (c) 1988, prime Computer, Inc.]
(creating mailbox for Jones...)
Error code 327 from routine crepw$
Error code 10 from routine ac$set
```

If you do not allow users to create password directories, you must create mailboxes for them. Without mailboxes, users can send mail but they cannot receive mail.

To create mailboxes for users, issue the `CREATE_MAILBOX` command. The `CREATE_MAILBOX` command has the following format:

```
CREATE_MAILBOX user_id
```

where *user_id* is the user ID of the user for whom you want to create a mailbox. If you omit *user_id*, you are prompted to enter one.

Note

The `CREATE_MAILBOX` command creates mailboxes as hashed directories. Password directories are not hashed directories. For a description of hashed directories, see the *Operator's Guide to File System Maintenance*.

The user for whom you create a mailbox must have at least PDALUR access rights to the MAILER*>MBOX directory. MAIL checks these rights before it creates a mailbox. A mailbox that is created has the following ACL rights:

```
MAILER_DAEMON: ALL
user_id: ALL
$REST: NONE
```

An error message is displayed if the user already has a mailbox.

Note

MAIL does not verify that the user ID that you specify is in the SAD. Therefore, if the user ID is not currently a valid user ID on the system, MAIL does not display an error message.

Activating PRIMOS TCP/IP

This chapter describes how to start PRIMOS TCP/IP software. It includes the following topics:

- TCP/IP processes and phantoms
- How to start TCP/IP after it has been installed on a 50 Series system
- How sockets are allocated among PRIMOS TCP/IP facilities after the product is started
- ACLs on the System Administration Directory (SAD)
- How to shut down PRIMOS TCP/IP
- How to modify the HOSTS.TXT and HOSTNAME_CONFIG files without stopping PRIMOS TCP/IP
- How to modify the PRIMOS.COMI file

PRIMOS TCP/IP Processes and Phantoms

After startup, PRIMOS TCP/IP consists of a number of processes that perform the following tasks:

- Handling remote FTP requests
- Forwarding MAIL messages
- Transferring files
- Monitoring other processes, file transfers, the LHC, and network traffic

These processes are described in the next sections.

Note

At Release 2.0, user FTP is implemented with PRIMOS socket library routines. The WSI_FTP_USER0 process, the WSI_USER_Phantomnn processes, and associated startup commands have been eliminated. See the section, The Allocation of Sockets, later in this chapter for a description of how sockets are allocated.

FTP Server Process

PRIMOS TCP/IP consists of one or two Server processes named `TCPFTP_SERVERnn`, where *nn* is an octal number that identifies the LHC it runs on. (Each LHC dedicated to PRIMOS TCP/IP has one Server process.) The Server process listens for a request from a user on a remote system. When the Server process detects a request from a remote user, it spawns a phantom named `TCPFTP_SERVER_PHANTOMnn`, where *nn* is an octal number that identifies the controller on which the phantom is running. The Server process can spawn more than one phantom but each phantom has the same name.

The Server process phantom (`TCPFTP_SERVER_PHANTOMnn`) performs the following file transfer functions for a user on a remote system who wants to send files to or retrieve files from the Prime host:

- Initiates the **data connection** (the communication path that is used to transfer data) between the remote system and the 50 Series host
- Transfers data in accordance with parameters that the remote user specifies

MAIL Servers

TCP/IP MAIL consists of the following three processes.

MAILER_DAEMON: The `MAILER_DAEMON` is responsible for managing mail requests that users on the 50 Series submit. If the destination of a mail message is another user on the same system, `MAILER_DAEMON` delivers the message that the user interface put in the output queue. `MAILER_DAEMON` is also responsible for delivering mail to a remote user whose system is connected to the local system by PRIMENET. If the destination is a user on a remote system connected to the LAN300, `MAILER_DAEMON` leaves the request in the output queue directory for the `SMTP_SENDER0` process. The `MAILER_DAEMON` runs on the host.

SMTP_SENDER0 Process: The `SMTP_SENDER0` process is responsible for delivering mail over the LAN300 and the Internet via SMTP. The `SMTP_SENDER0` process periodically checks the output queues and passes mail to the network for delivery. One `SMTP_SENDER0` process runs on each host.

SMTP_SERVERnn Process: Each LHC300 dedicated to PRIMOS TCP/IP has one `SMTP_SERVERnn` process. (*nn* is an octal number that identifies the LHC on which the process runs.) `SMTP_SERVERnn` receives mail from the network and puts it in the input queue directory. The `SMTP_SERVERnn` process runs on the host.

PRIMOS TCP/IP Manager

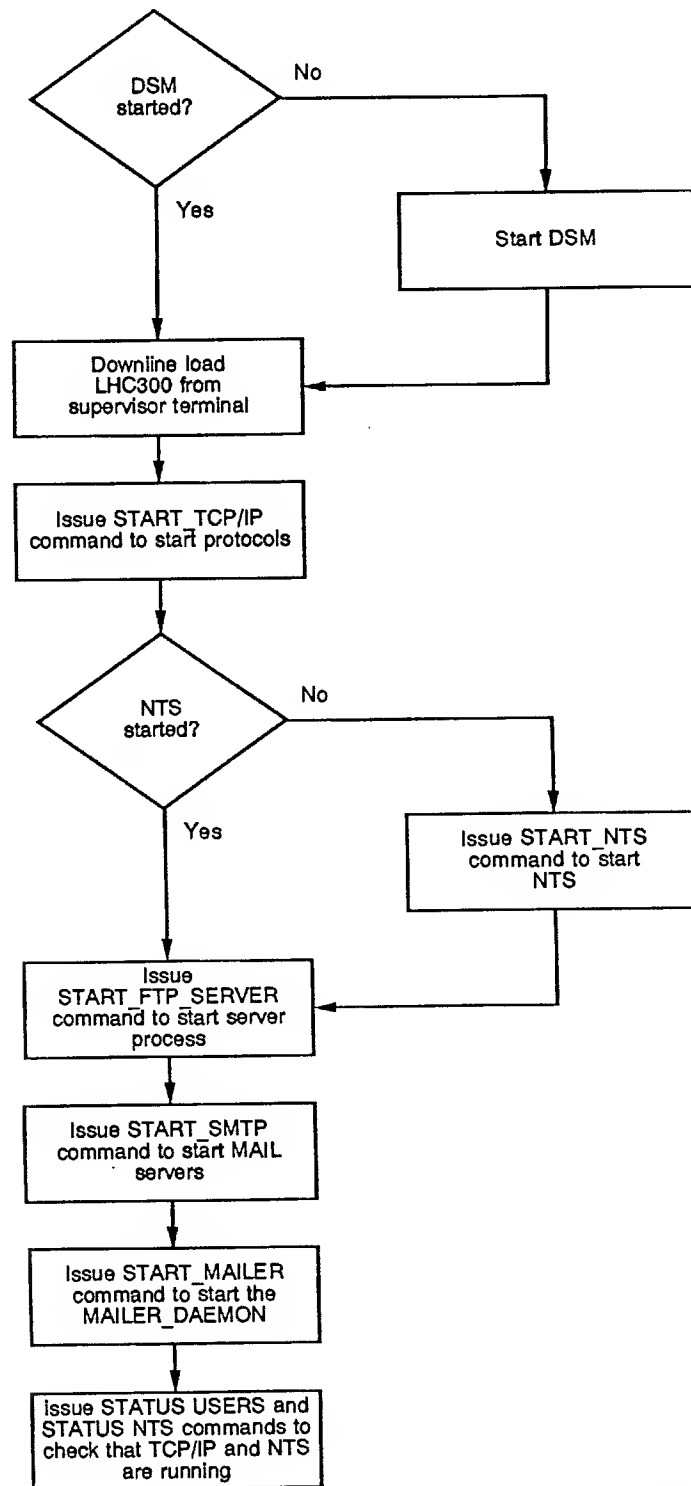
TCPIP_MANAGER is the TCP/IP managing process that is started when the START_TCP/IP command is issued. When PRIMOS TCP/IP is started, the TCPIP_MANAGER reads the TCP/IP*HOSTS file into memory. After the product starts, TCPIP_MANAGER keeps track of TCP/IP phantoms and monitors activity between the 50 Series host and the LHC. If the system is warm started or if the LHC fails, the managing process is responsible for

- Logging out all TCP/IP-related phantoms
- Reinitializing connections between the controller and host
- Restarting PRIMOS TCP/IP and restoring it to the same condition that existed before the warm start or LHC failure

Chapter 16, Monitoring PRIMOS TCP/IP, contains more information on PRIMOS TCP/IP recovery procedures.

Network Management Server

The Network Management server (NM_SERVER) is an independent process that runs on a 50 Series host after the START_TCP/IP or START_NTS command is issued. NM_SERVER is responsible for coordinating network management activity for that system. Figure 14-1 shows the sequence of steps to start PRIMOS TCP/IP. The next sections explain these steps.



Q10155-3LA-6-2

Figure 14-1
Starting PRIMOS TCP/IP

Starting PRIMOS TCP/IP

To start PRIMOS TCP/IP, perform the following steps from the supervisor terminal:

1. Downline load PRIMOS TCP/IP protocols onto the LHC
2. Start PRIMOS TCP/IP protocols on the LHC
3. Start NTS
4. Start PRIMOS FTP server process
5. Start the MAIL servers and the MAILER_DAEMON

Notes

In order to downline load PRIMOS TCP/IP protocols onto the LHC, you must have installed the Translator Family product (Release T1.0-21.0 or greater). Refer to the *Translator Family Software Release Document* for information on how to install the Translator Family.

The Distributed Systems Management (DSM) facility must be started and stable before you downline load an LHC, start PRIMOS TCP/IP, or start NTS. Events that originate on an LHC are sent to LAN300 Network Management software, which, in turn, logs them to DSM. Both PRIMOS TCP/IP and NTS also log events to DSM. Refer to the *DSM User's Guide* for instructions on starting DSM.

Downline Loading the LHC

This section describes how to downline load PRIMOS TCP/IP protocols onto the LAN Host Controller 300 (LHC).

To downline load one or more LHCs, you must use the `COMM_CONTROLLER` command, which has the following format:

`COMM_CONTROLLER subcommand [options]`

Only the System or Network Administrator can issue the `COMM_CONTROLLER` command from the supervisor terminal.

COMM_CONTROLLER Subcommands: The `COMM_CONTROLLER` command has the following subcommands that apply to PRIMOS TCP/IP:

▷ **-LOAD**

Specifies that the LHC (defined in the `-DEVICE_ADDRESS` and `-DEVICE` options) is to be initialized and downline loaded. The controller is automatically shut down, verified, and loaded.

▷ **-UPLINE_DUMP**
-ULD

Initiates a request that the intelligent controller (defined in the **-DEVICE_ADDRESS** and **-DEVICE** options) dump a memory image to the file specified in the **-PATHNAME** option.

▷ **-HELP**

Displays the text associated with any of the subcommands. This is the default if you omit one of the other main subcommands.

COMM_CONTROLLER Options: The following options to **COMM_CONTROLLER** subcommands are described in alphabetical order. You can place the options associated with the above subcommands anywhere on the command line, but the combination of options for a specified subcommand is critical.

▷ **-ALL**

Specifies that other **COMM_CONTROLLER** command options refer to all controllers of the type specified with the **-DEVICE** option. If you use the **-DEVICE_ADDRESS** and **-ALL** options in the same command string, the **COMM_CONTROLLER** command is rejected and an error message is displayed. You cannot use this option with the **-UPLINE_DUMP** subcommand. If a LAN300 is using some LHCs and PRIMOS TCP/IP is using other LHCs, you cannot use the **-ALL** option. The **-ALL** option causes the same downline load file to go to all LHC controllers.

▷ **-DEVICE LHC**
-DEV

Specifies that the command is addressed to an LHC type controller. You must use this option with the **-DEVICE_ADDRESS** option. This option is mandatory for all **COMM_CONTROLLER** command lines except the **-HELP** option.

▷ **-DEVICE_ADDRESS address**
-DA

Specifies the LHC controller device address to be downline loaded. The argument is a two-digit octal number entered in the **THISHOST** file.

If you enter an incorrect device address, an error message and a list of valid intelligent controller device addresses are displayed. You must use this option with the **-DEVICE** option.

- ▷ -NO_QUERY
-NQ

Suppresses the confirmation prompt (Continue?) and results in the appropriate action being carried out on the LHC regardless of the controller's current state. Use the -NO_QUERY option whenever you include the COMM_CONTROLLER command in a CPL program or the PRIMOS.COMI file.

- ▷ -PATHNAME *pathname*
-PN

Specifies the pathname of the controller downline load file. *pathname* consists of a maximum of 128 characters and must define an existing PRIMOS filename. The default PRIMOS TCP/IP downline loadfile is

DOWN_LINE_LOAD*>LHC300_TCP.DL

If you use *pathname* alone, the DOWN_LINE_LOAD* directory is assumed. You must specify a protocol with the -PROTOCOL option (described next). The protocol that you select informs the COMM_CONTROLLER command what download file to use. Therefore, when you specify a protocol, you can omit the -PATHNAME option.

- ▷ -PROTOCOL [*protocol*]
-PR

Specifies the protocol token combination downline loaded onto the LHC. Specify TCP for PRIMOS TCP/IP protocols as in the following example:

-PR TCP

You cannot downline load both the NTS and the PRIMOS TCP/IP protocols on the same LHC. For example, the command option -PROTOCOL NTS_TCP causes an error message to be displayed. If you use -PROTOCOL and -PATHNAME in the same command string, the -PROTOCOL option is rejected and an error message is displayed.

Note

You can run LAN300 on an LHC that has been running PRIMOS TCP/IP or run PRIMOS TCP/IP on an LHC that has been running LAN300; you must, however, cold start the system before you change protocols with the COMM_CONTROLLER command.

An Example of Using the COMM_CONTROLLER Command: To downline load the LHC for PRIMOS TCP/IP, issue the COMM_CONTROLLER command from the supervisor terminal, as shown in the following example:

OK, **COMM_CONTROLLER -LOAD -DEV LHC -DA 56 -NQ -PR TCP**
[COMM_CONTROLLER Rev. 22.0 Copyright (c) 1988 Prime Computer, Inc.]

LHC prom self-verify diagnostics in progress...

LHC downline load in progress...

LHC downline load operation results:
LHC at address 56: SUCCESSFUL

Starting PRIMOS TCP/IP on the Controller

After you downline load PRIMOS TCP/IP protocols onto the controller, you must start PRIMOS TCP/IP with the **START_TCP/IP** command. Issue the **START_TCP/IP** command from the supervisor terminal.

Note

The **START_TCP/IP** command fails unless both a **HOSTS** file and a **GATEWAYS_nn** file are in the **TCP/IP*** directory. You must run the Host Table Utility (**TCP/IPHTU**) before you issue the **START_TCP/IP** command. The **TCP/IPHTU** is described in Chapter 12, Configuring PRIMOS TCP/IP.

The **START_TCP/IP** command, which has no options and activates all the controllers that are running PRIMOS TCP/IP protocols, causes PRIMOS TCP/IP to respond with the following messages:

OK, **START_TCP/IP**

[**START_TCP/IP** Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
[Copyright (c) 1988, Bolt, Beranek & Newman Inc.]

PRIMOS TCP/IP Initialization Message.....
Processing Beginning.

PRIMOS TCP/IP Initialization Message.....
The TCP/IP Manager is being started.

PRIMOS TCP/IP Initialization Message.....
Network Management is being started for TCP/IP.
NMSr has started up network management for TCP/IP service.

TCP/IP Initialization Message.....
Successfully started Network Management for TCP/IP.

TCP/IP Initialization Message.....

The TCP/IP User Interface has been initialized.

TCP/IP Initialization Message.....

Processing Complete.

The START_TCP/IP command starts up the LAN300 Network Management server (NM_SERVER), which performs the following tasks:

- Starts LAN300 Network Management software if it has not been started
- Spawns a phantom named TCPIP_MANAGER that is the TCP/IP managing process
- Creates local log files named START_TCP/IP_LOG, MANAGER_LOG, and HOSTNAME_LOG in the TCP/IP*>LOG_FILES directory
- Passes Internet and gateway addresses to the controller

Note

You must execute the START_TCP/IP command before you issue any other TCP/IP startup command. You must execute the STOP_TCP/IP command (described in a later section) before you execute START_TCP/IP unless the system is being cold started.

Starting NTS

After you start the LHC, start NTS from the supervisor terminal with the following command:

START_NTS [*config_pathname*]

where *config_pathname* indicates the pathname of the NTS configuration file. The default pathname is NTS*>NTS.CONFIG. NTS responds with the following message:

OK, **START_NTS**

[START_NTS Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

NMSr has started up network management for NTS service.

Starting the PRIMOS FTP Server Process

After you start NTS, you can start the PRIMOS FTP server process from the supervisor terminal.

Note

At Release 2.0 of PRIMOS TCP/IP, user FTP is implemented with PRIMOS socket library routines. The WSI300 FTP User process has been eliminated.

Each LHC in the host has one Server process. (NTS may be started before or after the PRIMOS FTP server process. The starting order is not important.)

The command to activate an FTP server process has the following format:

```
START_FTP_SERVER [-DEVICE_ADDRESS dev_address] [-ALL] [-PROJECT]
                  -DA
```

where *dev_address* specifies the LHC device address listed in the THISHOST configuration file. You can use the `-ALL` option to specify that the `START_FTP_SERVER` command refers to all LHCs running PRIMOS TCP/IP. `-ALL` is the default.

`-PROJECT` specifies that users connecting to a 50 Series host are prompted to enter a project ID (in addition to a user ID and password). If you omit `-PROJECT`, users are not prompted to enter a project ID. No project prompt is the default.

Notes

When you omit `-PROJECT` from the command line to select the default, users are logged in to their own default projects. If you have not specified default projects with the `EDIT_PROFILE` utility, users cannot log in.

If you create a new project after you install TCP/IP, you must run the `TCP/IP.INSTALL_ACL.CPL` file to reset ACLs on the SAD. See Chapter 11, Installing PRIMOS TCP/IP, for a description of the `TCP/IP.INSTALL_ACL.CPL` file.

After you enter the command, PRIMOS TCP/IP responds with the following message:

```
OK, START_FTP_SERVER -PROJECT
[START_FTP_SERVER Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

Server FTP Initialization Message.....
Processing Beginning.

Server FTP Initialization Message.....
Spawning Server FTP.

Server FTP Initialization Message.....
Server FTP has been spawned.
Device Address is 56
```

The `START_FTP_SERVER` command performs the following tasks:

- Activates a phantom process named `TCPFTP_SERVERnn`, where *nn* is the device address (in octal) of the LHC on which the phantom is running
- Creates local log files named `PORTAL_LOG` and `SERVER_FTP_LOG` in the `TCP/IP*>LOG_FILES` directory

After the Server process starts, remote FTP users can submit requests. The Server process spawns phantoms named `TCPFTP_SERVER_PHANTOM nn` , where nn is the address of the controller in octal. These Server phantom processes service remote users.

Starting PRIMOS TCP/IP MAIL

After you start the FTP server process, start the TCP/IP MAIL servers: `SMTP_SENDER0`, `SMTP_SERVER nn` , and `MAILER_DAEMON`. `SMTP_SENDER0` transfers messages between the input and output queues and a LAN300. `SMTP_SERVER nn` receives messages from the network. (nn is the device address of the controller.) The `MAILER_DAEMON` sends messages over PRIMENET and transfers mail between users on the local host. Each host has one `MAILER_DAEMON` and `SMTP_SENDER0` process. Each LHC has one `SMTP_SERVER` process.

Starting `SMTP_SENDER0` and `SMTP_SERVER nn` : The command to activate the MAIL servers has the following format:

```
START_SMTP [-DEVICE_ADDRESS dev_address] [-ALL]
           -DA
```

where *dev_address* specifies the LHC device address listed in the `THISHOST` configuration file. You can use the `-ALL` option to specify that the `START_SMTP` command refers to all LHCs running PRIMOS TCP/IP. `-ALL` is the default.

After you enter the command, PRIMOS TCP/IP responds with the following message:

```
OK, START_SMTP -DA 56
[START_SMTP Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

SMTP Initialization Message.....
  Processing Beginning.

SMTP Initialization Message.....
  Starting Sender & Servers.

SMTP Initialization Message.....
  SMTP Server has been started.
  Device Address is 56

SMTP Initialization Message.....
  SMTP SENDER has been spawned.
  Device Address is 56
```

The `START_SMTP` command performs the following tasks:

- Activates `SMTP_SENDER0` and `SMTP_SERVER nn` processes described above
- Creates a local log file named `SMTP_LOG` in the `TCP/IP*>LOG_FILES` directory

Starting the MAILER_DAEMON: The following command starts the MAILER_DAEMON:

START_MAILER

After you enter the START_MAILER command, PRIMOS TCP/IP responds with the following message:

```
OK, START_MAILER
[START_MAILER Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
Mailer Daemon started up.
```

After the MAILER_DAEMON process starts, users can send and receive mail messages.

Note

You must create a MAIL configuration file before you start the MAILER_DAEMON. See Chapter 13, Configuring PRIMOS TCP/IP MAIL, for a description of the configuration file.

Process and Phantom Names

This section lists the names of phantoms and processes that are running after you start PRIMOS TCP/IP, NTS, and LAN300 Network Management.

NM_SERVER

LAN300 Network Management server activated by the START_TCP/IP command

TCPIP_MANAGER

TCP/IP managing process activated by the START_TCP/IP command

UBI_SERVER

A process activated by the START_TCP/IP command and required by the socket library

TCPFTP_SERVER_{nn}

PRIMOS TCP/IP server FTP process activated by the START_FTP_SERVER command

TCPFTP_SERVER_PHANTOM_{nn}

A phantom spawned by the PRIMOS TCP/IP server process to service FTP requests of remote users

MAILER_DAEMON

MAIL server activated by the START_MAILER command

SMTP_SENDER0

MAIL server activated by the START_SMTP command

SMTP_SERVERnn

MAIL server activated by the START_SMTP command

The following example illustrates the screen output of the STATUS USERS command after you start PRIMOS TCP/IP:

OK, *STAT USERS*

User	No	Line oct (dec)	Devices
SYSTEM	1	asr	<EXAMP14>
NTS_SERVER	46	ncm	<EXAMP14>
TIMER_PROCESS	48	kernel	<EXAMP14>
LOGIN_SERVER	49	LSr	<EXAMP14> (3)
DSMSR	50	DSM	<EXAMP14>
DSMASR	51	DSM	<EXAMP14>
SYSTEM_MANAGER	54	SMSr	<EXAMP14>
NM_SERVER	55	phant	<EXAMP14>
TCPIP_MANAGER	56	phant	<EXAMP14>
TCPFTP_SERVER56	57	phant	<EXAMP14>
MAILER_DAEMON	58	phant	<EXAMP14>
SMTP_SENDER0	59	phant	<EXAMP14>
SMTP_SERVER56	60	phant	<EXAMP14>
ISC_NETWORK_SERVER	61	ISCNsr	<EXAMP14>
UBI_SERVER	62	kernel	<EXAMP14> (5)

Issue the STATUS NTS command to be certain that NTS is running. The STATUS NTS command has the following screen output after NTS is started:

OK, *STATUS NTS*

NTS is currently started

NTS config file <TOP>NTS*>NTS.CONFIG

The Allocation of Sockets

User FTP, user TELNET, and MAIL (SMTP) are implemented with PRIMOS socket library routines. PRIMOS TCP/IP can support a maximum of 32 FTP, user TELNET, or socket users simultaneously.

After PRIMOS TCP/IP is started, the sockets are allocated among PRIMOS TCP/IP facilities in the following manner:

- MAIL uses two sockets.
- A user of TELNET uses one socket.
- A user of FTP uses one socket.
- The TCPIP_MANAGER uses one socket if the Hostname Service is configured.
- A process that calls the PRIMOS socket library can use a maximum of 32 sockets simultaneously for each installed LHC.

How to Update the HOSTS.TXT or HOSTNAME_CONFIG File

You can update the HOSTS.TXT file or HOSTNAME_CONFIG file (but not the THISHOST file) without stopping and restarting PRIMOS TCP/IP. When the TCPIP_MANAGER is started, it reads the HOSTS table (created by the Host Table Utility) into memory. The information read into memory can be modified by using the TCP/IP_INIT_HOSTS command described in this section.

After PRIMOS TCP/IP is started, perform the following steps to update the configuration files:

1. Update TCP/IP*>HOSTS.TXT with EMACS or ED.
2. Update the Hostname Service configuration parameters in the TCP/IP*>HOSTNAME_CONFIG file.
3. Run the TCP/IP Host Table Utility (TCPHTU) to create a new version of TCP/IP*>HOSTS.
4. Issue the TCP/IP_INIT_HOSTS command to force the changes to take effect.

The next section describes the TCP/IP_INIT_HOSTS command.

TCP/IP_INIT_HOSTS Command

The TCP/IP_INIT_HOSTS command forces the TCP/IP_MANAGER to initialize its internal host table by reading the current version of TCP/IP*>HOSTS. The HOSTNAME_CONFIG file is also read so that you can change the parameters that control the operation of the Hostname Service protocol.

The TCP/IP_INIT_HOSTS command has the following format:

TCP/IP_INIT_HOSTS $\left[\left\{ \begin{array}{l} -\text{SAVE_CACHE} \\ -\text{NO_SAVE_CACHE} \end{array} \right\} \right]$

Use the command option to override the setting of the save cache parameter in the HOSTNAME_CONFIG file. The -SAVE_CACHE option causes the PRIMOS TCP/IP Manager to save cache entries when you issue either a STOP_TCP/IP or TCP/IP_INIT_HOSTS

command. The `-NO_SAVE_CACHE` causes the PRIMOS TCP/IP Manager to discard cache entries when you issue either the `STOP_TCP/IP` or `TCP/IP_INIT_HOSTS` command. `-SAVE_CACHE` is the default.

Saved cache entries are stored in a binary file named `TCP*>HOSTNAME_CACHE`.

Shutting Down PRIMOS TCP/IP

This section describes how to shut down PRIMOS TCP/IP.

STOP_TCP/IP Command

If you are required to stop the PRIMOS FTP and MAIL Server processes, enter the `STOP_TCP/IP` command from the supervisor terminal. The `STOP_TCP/IP` command generates the following program messages:

```
OK, STOP_TCP/IP
[STOP_TCP/IP Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]
REALLY? YES

PRIMOS TCP/IP Termination Message.....
    Processing Beginning.

PRIMOS TCP/IP Termination Message.....
    Shutting down TCP/IP.

PRIMOS TCP/IP Termination Message.....
    Successfully shutdown TCP/IP.

TCP/IP Termination Message.....
    Shutting down Network Management for TCP/IP.
NMSr is shutting down TCP/IP service support.

TCP/IP Termination Message.....
    Successfully shutdown PRIMOS TCP/IP.
    Be sure to downline load ALL TCP/IP controllers before restarting!
```

The `STOP_TCP/IP` command performs the following tasks:

- Stops the FTP server process
- Stops the MAIL servers (`SMTP_SENDER0` and `SMTP_SERVERnn`)
- Creates a log file named `STOP_TCP/IP_LOG` in the `TCP/IP*>LOG_FILES` directory

- Logs events to the DSM log (TCP/IP*>DSM_LOGFILE) and the STOP_TCP/IP_LOG file
- Notifies the TCP/IP manager process (TCPIP_MANAGER), which logs out PRIMOS TCP/IP-related processes and then logs itself out

When you stop TCP/IP, the TCPIP_MANAGER may not log out immediately. You can, however, downline load the LHC300 after you issue the STOP_TCP/IP command even if the STATUS USERS command shows that the TCPIP_MANAGER or any other TCP/IP-related process is running.

STOP_MAILER Command

To stop the MAILER_DAEMON, issue the STOP_MAILER command.

Note

Once started, the UBI_SERVER and the NTS_SERVER (provided the value of NTSUSR is greater than zero) run until the system is shutdown.

Modifying the PRIMOS.COMI File

This section describes PRIMOS TCP/IP-related commands that you can insert in PRIMOS.COMI. PRIMOS.COMI is the system startup file that is executed at cold start. If you modify the PRIMOS.COMI file, you must put the commands in the following order:

```
START_DSM
START_MAILER
COMM_CONTROLLER
START_TCP/IP
START_FTP_SERVER
START_SMTP
START_NTS
```

Note

All PRIMOS TCP/IP startup commands (except START_MAILER) must appear after the COMM_CONTROLLER command in the PRIMOS.COMI file. Any command that spawns a phantom, which, in turn, spawns another phantom must appear after the MAXUSR command. The COMM_CONTROLLER command must appear after the MAXUSR command.

START_DSM Entry

The START_DSM command starts the Distributed Systems Management (DSM) facility on the local system. In the PRIMOS.COMI file, the START_DSM command must come after the CONFIG command but before the COMM_CONTROLLER and other PRIMOS TCP/IP-related commands to ensure that PRIMOS TCP/IP and NTS messages are logged. DSM must be running before LAN300 Network Management software is started.

The START_DSM command entry in PRIMOS.COMI has the following format:

```
START_DSM
```

COMM_CONTROLLER Entry

The COMM_CONTROLLER command loads software into the LHC300 controllers in the system. You must insert this command in PRIMOS.COMI before the START_TCP/IP command or START_NTS command. If the LHCs in the system are not running the same software and protocols, you must issue a separate COMM_CONTROLLER command for each LHC. Remember to use the -NQ option to enable you to execute the command without being prompted for confirmation. Refer to the section titled Downline Loading the LHC earlier in this chapter for the syntax of this command.

START_TCP/IP Entries

The START_MAILER entry starts the MAILER_DAEMON. The START_TCP/IP command starts the LHC(s) with PRIMOS TCP/IP protocols and the LAN300 Network Management facility on the host. Place this command near the beginning of PRIMOS.COMI but after the COMM_CONTROLLER and START_DSM commands to give PRIMOS TCP/IP adequate time to initialize itself on the LHC. The START_FTP_SERVER and START_SMTP commands start the PRIMOS FTP server and the PRIMOS MAIL processes (SMTP_SENDER0 and SMTP_SERVERnn), respectively.

START_NTS Entry

The START_NTS command starts the Network Terminal Service. START_NTS has the following format:

```
START_NTS [config_pathname]
```

where *config_pathname* indicates the pathname of the NTS configuration file. The default pathname is NTS*>NTS.CONFIG.

An Example of PRIMOS TCP/IP Entries in PRIMOS.COMI

The following example shows the MAXUSR command and PRIMOS TCP/IP entries in a PRIMOS.COMI file.

```
START_MAILER /* Start the MAILER_DAEMON
MAXUSR
COMM_CONTROLLER -LOAD -DEV LHC -DA 56 -PR TCP -NQ
/* Downline load LHC with PRIMOS TCP/IP protocols
START_TCP/IP          /* Start TCPIP_MANAGER process
START_FTP_SERVER -DA 56 /* Start TCPFTP_SERVER process
START_SMTP -DA 56 /* Start the MAIL servers
```

Sample PRIMOS TCP/IP Configurations

This chapter presents four sample PRIMOS TCP/IP configurations and describes the following for each:

- PRIMOS TCP/IP configuration file entries
- THISHOST file entries
- HOSTNAME_CONFIG file entries
- LHC directives
- CONFIG_NTS commands and screens
- COMM_CONTROLLER command

Read Chapters 11, 12, and 14, which describe how to install, configure, and activate PRIMOS TCP/IP, before you read this chapter.

Sample Network 1

The network in Example 1 is a Class B network with six systems attached to it. See Figure 15-1. Its network number is 129.42.0.0 and its network name is EXAM1NET. (Both the network number and name are arbitrary designations that follow the conventions described in Chapter 12, Configuring PRIMOS TCP/IP.) SYS1, SYS2, and SYS3 are Prime systems. NPRIME1, NPRIME2, and NPRIME3 are other vendors' systems. The network does not implement subnetting.

HOSTS.TXT File

For each 50 Series system, you must create a configuration file named HOSTS.TXT that resides in the TCP/IP* directory. (You must also configure the other vendors' systems according to the instructions in their documentation.) The HOSTS.TXT file on all three 50 Series systems is identical, as shown in the following example:

```
NET : 129.42.0.0 : EXAM1NET :  
HOST : 129.42.0.1 : SYS1 : 9955 : PRIMOS : TCP/FTP, TCP/TELNET :  
HOST : 129.42.0.2 : SYS2 : 9750 : PRIMOS : TCP/FTP, TCP/TELNET :  
HOST : 129.42.0.3 : EN-SYS3 : 2350 : PRIMOS : TCP/FTP, TCP/TELNET :  
HOST : 129.42.0.4 : NPRIME1 : WHIZTRONICS 100 : WHIZ-OP : TCP/FTP, TCP/TELNET :  
HOST : 129.42.0.5 : NPRIME2 : WS-1200 : WS-OP : TCP/FTP, TCP/TELNET :  
HOST : 129.42.0.6 : NPRIME3 : WS-1400 : WS-OP : TCP/FTP, TCP/TELNET :
```

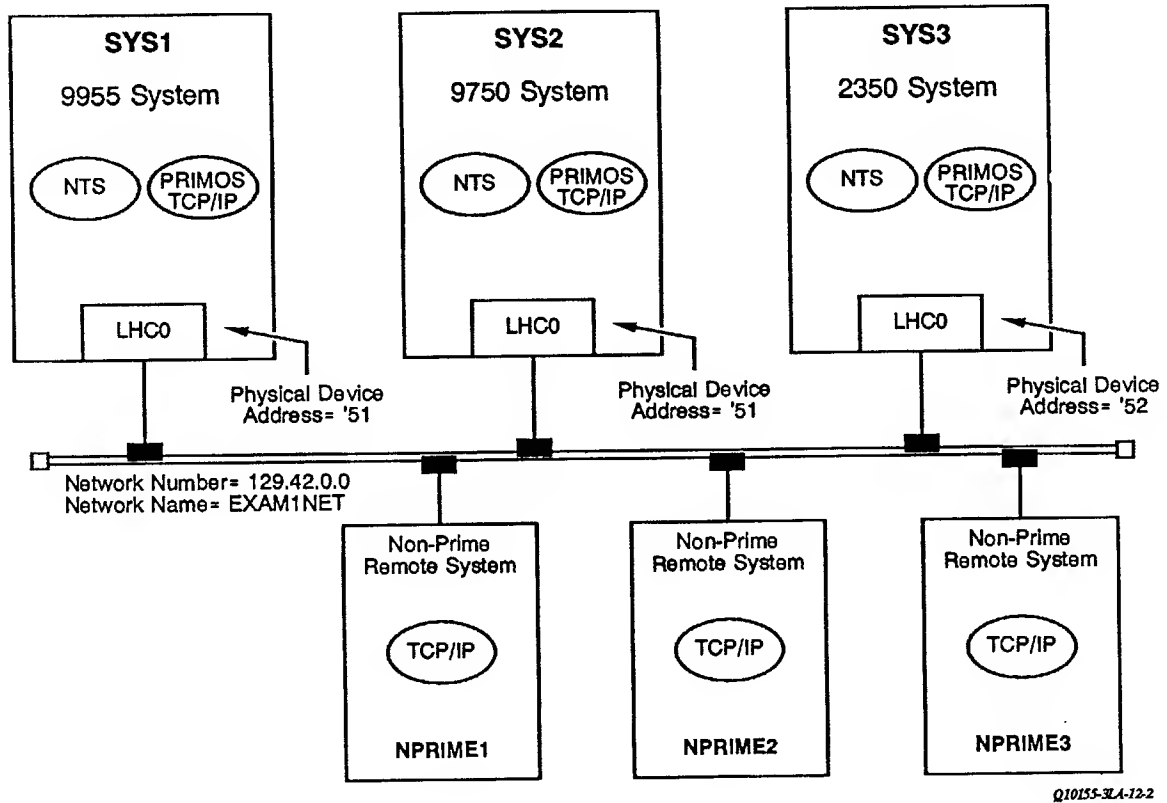


Figure 15-1
Sample Network 1

LHC Directive

Each 50 Series system has one LHC. The physical device address for the LHCs in SYS1 and SYS2 is 51₈. EN-SYS3 has the controller on device address 52₈. Therefore, the system configuration file for each system has the following entries for the LHC directive:

System	LHC Directive
SYS1	LHC 0 51
SYS2	LHC 0 51
EN-SYS3	LHC 0 52

THISHOST File

The SYSNAM directive in the system configuration files of SYS1, SYS2, and SYS3 sets the names of these systems at cold start as SYS1, SYS2, and EN.SYS3, respectively. SYS3 has a PRIMENET node name (EN.SYS3) that cannot be changed because users are accustomed to that

name. However, the name is not valid for PRIMOS TCP/IP under DoD standards because it contains a period. For PRIMOS TCP/IP the name is changed to EN-SYS3 in the THISHOST file. SYS1 and SYS2 can use the default system name in the THISHOST file.

The THISHOST file in the TCP/IP* directory of SYS1 has the following entry:

```
SYS1
device address = 51 : internet address = 129.42.0.1 :
```

The THISHOST file in the TCP/IP* directory of SYS2 has the following entry:

```
SYS2
device address = 51 : internet address = 129.42.0.2 :
```

The THISHOST file in the TCP/IP* directory of EN-SYS3 has the following entry:

```
EN-SYS3
device address = 52 : internet address = 129.42.0.3 :
```

HOSTNAME_CONFIG File

SYS1, SYS2, and EN-SYS3 have the following entries in their HOSTNAME_CONFIG files:

```
CACHE AGE = 25
SAVE CACHE = FALSE
SERVER ADDRESS = 129.42.0.4
```

The server address is the Internet address of NPRIME1. NPRIME1 supports the Hostname Service protocol.

CONFIG_NTS Session

The following listing is from the CONFIG_NTS session done for Sample Network 1. It describes a configuration file that is installed on all three 50 Series systems. It does not include any LTSs because none are on the network. Note that you do not configure other vendors' systems to look like LTSs. You must specify that unconfigured nodes are permitted. You create this listing by entering 6 (display, list or spool configuration) on the CONFIG_NTS main menu after you have created and saved the configuration. Also, note that CONFIG_NTS requires that the name of a system match the name entered with the SYSNAM directive. Therefore, the third 50 Series system is listed as EN.SYS3. See Figure 15-2.

```

CONFIG_NTS, revision 22.0                                Dump of configuration file:
                                                         <OURDISK>ADMIN>EXAM1_CONFIG.CONFIG

Sample configuration #1

File created on 16 Sep 86 at 13:29 by ADMIN
Last edited on 16 Sep 86 at 16:45 by ADMIN

- - - LAN: EXAM1 - - -

Physical LAN type: IEEE 802.3 10M 500 meter LAN
Unconfigured nodes are permitted.
Network management functions are not configured.

Host name          LHCs
-----
SYS1               LHC00
SYS2               LHC00
EN.SYS3            LHC00

No LTSs are configured.

```

Figure 15-2
Summary of Configuration File for EXAM1NET

The COMM_CONTROLLER Command

From the supervisor terminal of each 50 Series system, issue the following COMM_CONTROLLER command to downline load the LHCs for TCP/IP:

<i>System</i>	<i>Command</i>
SYS1	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 51 -PR TCP_TEL
SYS2	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 51 -PR TCP_TEL
EN-SYS3	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 52 -PR TCP_TEL

Sample Network 2

Sample network 2 is a Class C network with one 50 Series system (SYS1) and three other vendors' systems (NPRIME1, NPRIME2, NPRIME3) attached to it. (See Figure 15-3.) The name of the network is EXAM2NET and its number is 193.44.5.0. The network has two LTSs on the network. It does not use subnetting.

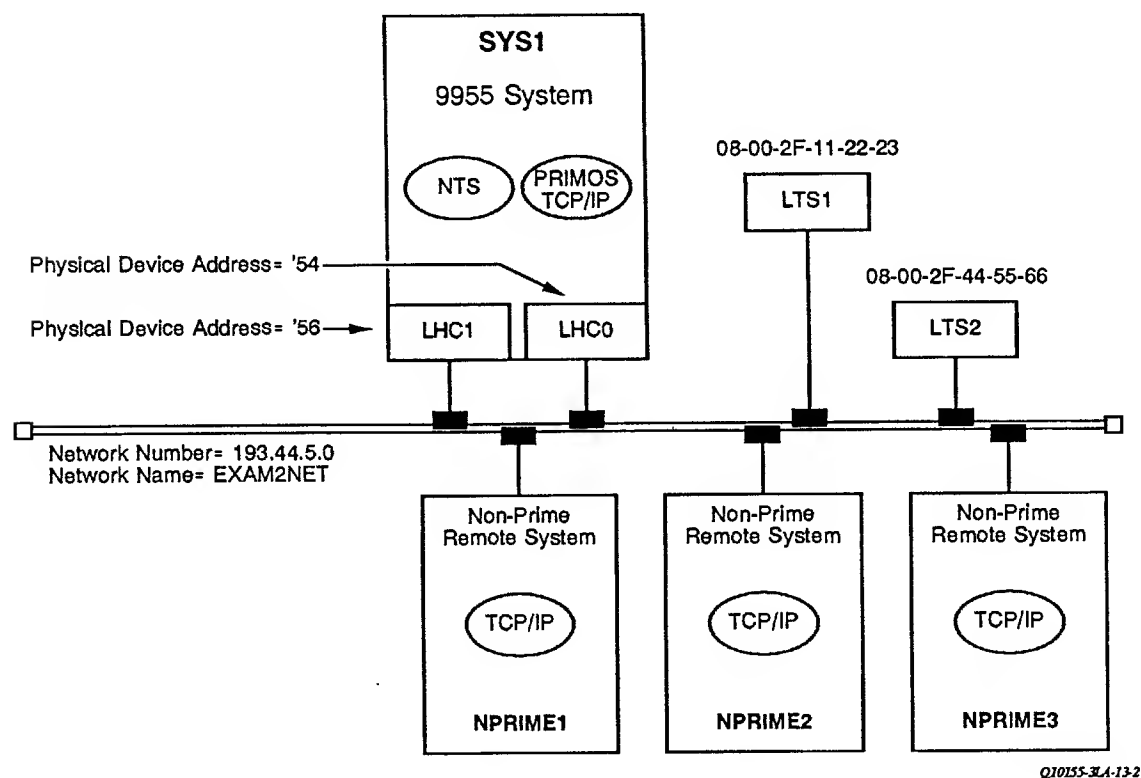


Figure 15-3
Sample Network 2

HOSTS.TXT File

You configure the 50 Series system by creating a file named HOSTS.TXT that resides in the TCP/IP* directory. Configure the other vendors' systems according to the instructions in their documentation. The HOSTS.TXT file on SYS1 has the following entries:

```
NET : 193.44.5.0 : EXAM2NET :
HOST : 193.44.5.1 : SYS1 : 9955 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 193.44.5.2 : NPRIME1 : WHIZTRONICS 100 : WHIZ-OP : TCP/FTP, TCP/TELNET :
HOST : 193.44.5.3 : NPRIME2 : WS-1200 : WS-OP : TCP/FTP, TCP/TELNET :
HOST : 193.44.5.4 : NPRIME3 : WS-1400 : WS-OP : TCP/FTP, TCP/TELNET :
```

LHC Directives

SYS1 has two LHC controllers. PRIMOS TCP/IP uses one controller (LHC00) to enable communication to the other vendors' hosts. NTS uses the other controller (LHC01) for communicating to the LTSs. PRIMOS TCP/IP does not communicate with the LTSs but uses NTS to provide TELNET service to terminal users on the other vendors' systems. Therefore, PRIMOS TCP/IP must be linked to one LHC controller and NTS must be linked to both.

PRIMOS TCP/IP uses LHC00, which has device address 54_g. NTS uses LHC01 with device address 56_g to communicate with the LTSs. Therefore, SYS1's system configuration file has the following entries for the LHC directive:

<i>System</i>	<i>LHC Directive</i>
SYS1	LHC 0 54
SYS1	LHC 1 56

Information about LHC01 is required in the system configuration file (CONFIG), but it is not required in TCP/IP-related configuration files. CONFIG_NTS associates LHC01 with the LTSs on the network.

THISHOST File

SYS1 is the name set for SYS1 at cold start. The THISHOST file can use the default system name. The THISHOST file in the TCP/IP* directory of SYS1 has the following entry:

```
;  
device address = 54 : internet address = 129.42.0.1 :
```

HOSTNAME_CONFIG File

SYS1 has the following entries in its HOSTNAME_CONFIG file:

```
CACHE AGE = 25  
SAVE CACHE = FALSE  
SERVER ADDRESS = 193.44.5.2
```

The server address is the Internet address of NPRIME1.

CONFIG_NTS Session

Figure 15-4 is from the CONFIG_NTS session for Sample Network 2.

Note that you can configure network management functions in this example. Network management services are provided to the LTSs from LHC01. LHC00 manages the connection to the other vendors' systems and LHC01 manages the connection to the LTSs and provides the downline load, upline dump, and event reporting services. However, you do not have to specify this controller information to CONFIG_NTS. NTS software knows which controller to use. In this example, unconfigured nodes are permitted so that the users on other vendors' systems can log in to the 50 Series system with PRIMOS TCP_IP.


```

CONFIG_NTS, revision 22.0                Dump of configuration file:
                                         <OURDISK>ADMIN>EXAM2_CONFIG.CONFIG

```

```

Sample configuration #2

```

```

File created on 16 Sep 88 at 16:52 by ADMIN
Last edited on 16 Sep 88 at 16:56 by ADMIN

```

```

- - - LAN: EXAM2 - - -
test configuration

```

```

Physical LAN type: IEEE 802.3 10M 500 meter LAN
Unconfigured nodes are permitted.

```

```

Network management functions:

```

	Primary	Secondary
LTS Downline Load	SYS1	
LTS Upline Dump	SYS1	
LTS Event Reporting	SYS1	

Host name	LHCs
SYS1	LHC00 LHC01

LTS name	Address
LTS1	08-00-2F-11-22-33
LTS2	08-00-2F-44-55-66

Figure 15-4
Summary of Configuration File for EXAM2NET

The COMM_CONTROLLER Command

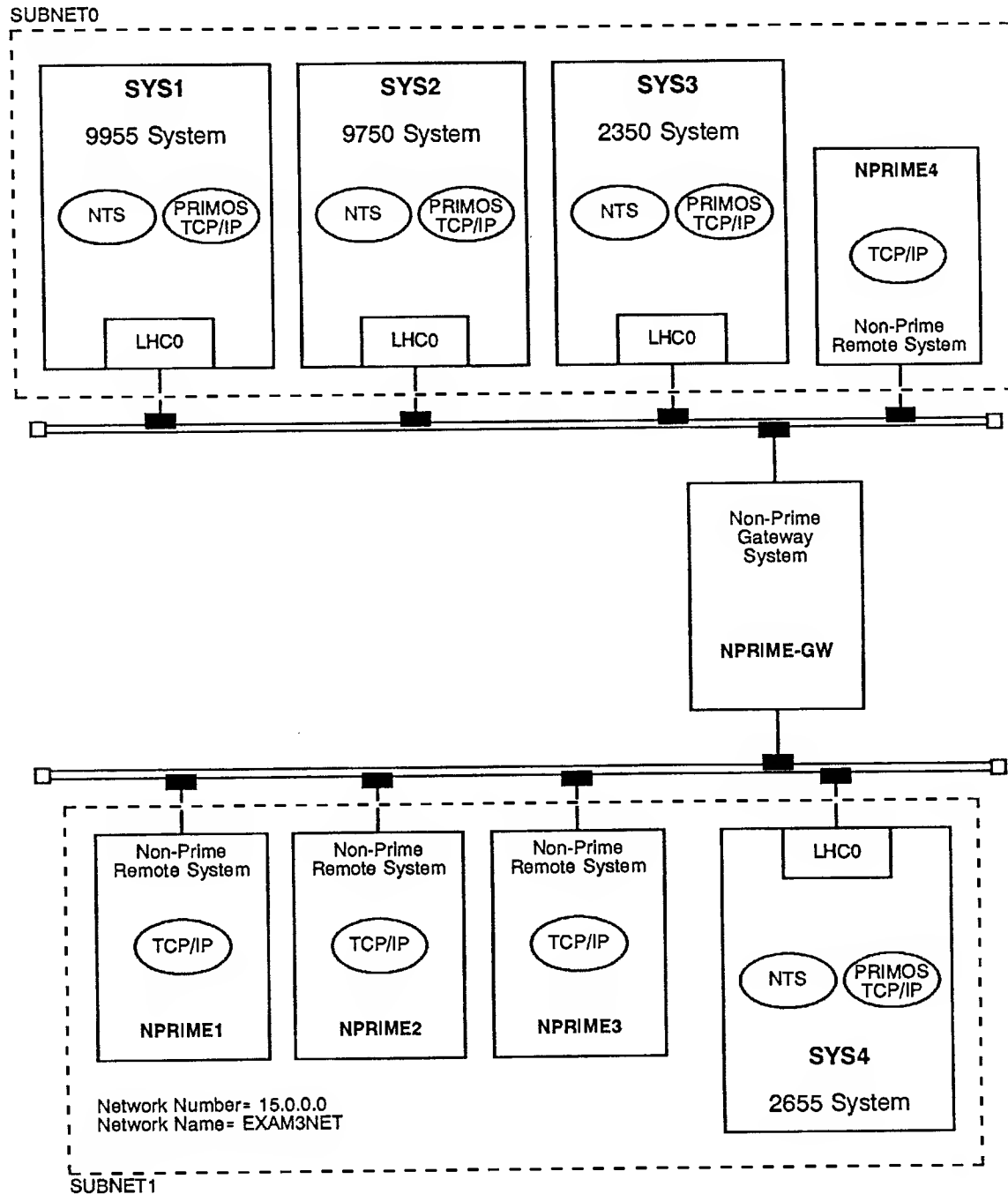
From the supervisor terminal, use the following commands to downline load the the LHCs for PRIMOS TCP/IP and NTS:

```
OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 54 -PR TCP_TEL
```

```
OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 56 -PR NTS
```

Sample Network 3

Sample Network 3 is a Class A network. The network number is 15.0.0.0 and its name is EXAM3NET. Of the eight systems on the network, four are 50 Series systems (SYS1, SYS2, EN-SYS3 and SYS4) and four are other vendors' systems (NPRIME1, NPRIME2, NPRIME3, and NPRIME4). The network has two subnetworks. Three 50 Series systems and one non-Prime are in one subnet; three other vendors' systems and one 50 Series system are in the other subnet. See Figure 15-5.



Q10255-3.4-14.2

Figure 15-5
Sample Network 3

HOSTS.TXT File

Configure each 50 Series system by creating a file named HOSTS.TXT that resides in the TCP/IP* directory on each system. Configure the other vendors' systems according to the instructions in their documentation. The HOSTS.TXT file on all four 50 Series systems is identical and has the following entries:

```
NET : 15.0.0.0 : EXAM3NET :
GATEWAY : 15.16.0.5, 15.0.0.5 : NPRIME-GW : WS-OP : TCP/FTP, TCP/TELNET :
HOST : 15.0.0.1 : SYS1 : 9955 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 15.0.0.2 : SYS2 : 9750 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 15.0.0.3 : EN-SYS3 : 2350 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 15.0.0.4 : NPRIME4 : WS-100 : WS-OP : TCP/FTP, TCP/TELNET :
HOST : 15.16.0.1 : NPRIME1 : WS-100 : WS-OP : TCP/FTP, TCP/TELNET :
HOST : 15.16.0.2 : NPRIME2 : WS-1200 : WS-OP : TCP/FTP, TCP/TELNET :
HOST : 15.16.0.4 : SYS4 : 2655 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 15.16.0.5 : NPRIME : WS-OP : TCP/FTP, TCP/TELNET :
```

Three 50 Series systems and one other vendor's system are on subnet 0. (The subnet number is contained in the leftmost four bits of the second octet.) The systems on subnet 1 have 16 in the second octet. Sixteen (decimal) consists of a one bit in the low order subnet bit. Sixteen decimal is 20 octal or 10 hex (00010000) with 1 in the rightmost bit of the subnet mask.

LHC Directive

Each 50 Series system has one LHC controller. The physical device address for the LHCs in SYS1 and SYS2 is 32₈. SYS3 and SYS4 have the controller on device address 37₈. Therefore, the system configuration file for each system has the following entries for the LHC directive:

<i>System</i>	<i>LHC Directive</i>
SYS1	LHC 0 32
SYS2	LHC 0 32
EN-SYS3	LHC 0 37
SYS4	LHC 0 37

THISHOST File

SYS1, SYS2, and SYS4 are the names set for these systems at cold start so that the THISHOST file can use the default system name. SYS3 is named EN.SYS3 at cold start. EN.SYS3 is a PRIMENET node name that cannot be changed without inconveniencing users. The PRIMENET name has a period in it, so PRIMOS TCP/IP cannot use the name. The name EN.SYS3 is changed to EN-SYS3 for PRIMOS TCP/IP.

The subnet mask is FFF00000 (in hex). The leftmost four bits of the second octet of the Internet address specifies the subnet. As in all Class A networks, the network number is contained in the first octet of the address.

The THISHOST file in the TCP/IP* directory of SYS1 has the following entry:

```
;
device address = 32 : internet address = 15.0.0.1 : subnet mask = FFF00000 :
```

The THISHOST file in the TCP/IP* directory of SYS2 has the following entry:

```
;
device address = 32 : internet address = 15.0.0.2 : subnet mask = FFF00000 :
```

The THISHOST file in the TCP/IP* directory of SYS3 has the following entry:

```
EN-SYS3
device address = 37 : internet address = 15.0.0.3 : subnet mask = FFF00000 :
```

The THISHOST file in the TCP/IP* directory of SYS4 has the following entry:

```
;
device address = 37 : internet address = 15.16.0.4 : subnet mask = FFF00000 :
```

HOSTNAME_CONFIG File

SYS1, SYS2, SYS3, and SYS4 have the following entries in their HOSTNAME_CONFIG files:

```
CACHE AGE = 25
SAVE CACHE = FALSE
SERVER ADDRESS = 15.16.0.2
```

The server address is the Internet address of NPRIME3.

CONFIG_NTS Session

CONFIG_NTS is not concerned with subnetting. Therefore, you do not enter any information in CONFIG_NTS about subnetting. Also, note that CONFIG_NTS requires that a system's name match its name entered with the SYSNAM directive. EN-SYS3 is listed as EN.SYS3. Figure 15-6 is the listing from the CONFIG_NTS session done for Sample Network 3.

```

CONFIG_NTS, revision 22.0                                Dump of configuration file:
                                                         <II>ADMIN>EXAM3_CONFIG.CONFIG

Sample configuration #3

File created on 18 Sep 86 at 11:31 by ADMIN

      - - - LAN: EXAM3 - - -

Physical LAN type: IEEE 802.3 10M 500 meter LAN
Unconfigured nodes are permitted.
Network management functions are not configured.

Host name          LHCs
-----
SYS1               LHC00
SYS2               LHC00
EN.SYS3            LHC00
SYS4               LHC00

No LTSSs are configured.

```

Figure 15-6
Summary of Configuration File for EXAM3NET

The COMM_CONTROLLER Command

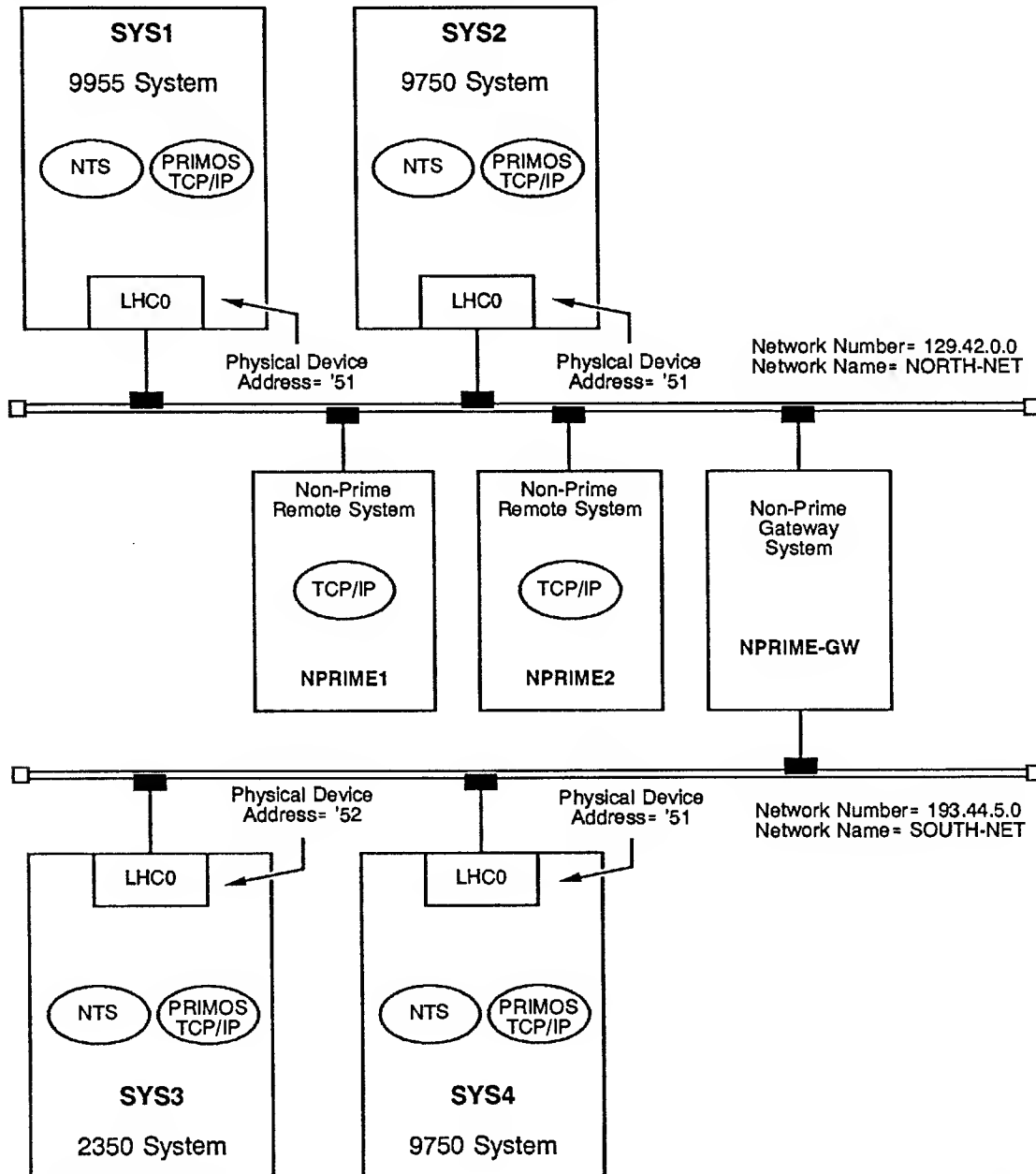
From the supervisor terminal of each 50 Series system, issue the following COMM_CONTROLLER commands to downline load the LHCs for PRIMOS TCP/IP:

<i>System</i>	<i>Command</i>
SYS1	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 32 -PR TCP_TEL
SYS2	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 32 -PR TCP_TEL
EN-SYS3	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 37 -PR TCP_TEL
SYS4	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 37 -PR TCP_TEL

Sample Network 4

Sample Network 4 consists of a Class B network connected to a Class C network by means of a gateway system. The Class B network (NORTH-NET) has four systems attached to it. The Class C network (SOUTH-NET) has two systems connected to it. The gateway system is connected to both networks.

The network numbers of NORTH-NET and SOUTH-NET are 129.42.0.0 and 193.44.5.0, respectively. SYS1, SYS2, SYS3, and SYS4 are Prime systems. NPRIME1, NPRIME2, and NPRIME-GW are other vendors' systems. Neither network implements subnetting. See Figure 15-7.



Q10155-3A-15-2

Figure 15-7
Sample Network 4

HOSTS.TXT File

For each 50 Series system, you must create a configuration file named HOSTS.TXT that resides in the TCP/IP* directory. (You must also configure the other vendors' systems according to the instructions in their documentation.) The HOSTS.TXT file on all four 50 Series systems is identical, as shown in the following example:

```
NET : 129.42.0.0 : NORTH-NET :
NET : 193.44.5.0 : SOUTH-NET :
GATEWAY : 129.42.0.1, 193.44.5.3: NPRIME-GW : XYZ-200 : : TCP/FTP :
HOST : 129.42.0.1: NPRIME : XYZ-200 : : TCP/FTP :
HOST : 129.42.0.2 : SYS1 : 9955 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 129.42.0.3 : SYS2 : 9750 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 129.42.0.4 : NPRIME1 : WS-1200 : WS-OP : TCP/FTP, TCP/TELNET :
HOST : 129.42.0.5 : NPRIME2 : WS-1400 : WS-OP : TCP/FTP, TCP/TELNET :
HOST : 193.44.5.1 : SYS3 : 2350 : PRIMOS : TCP/FTP, TCP/TELNET :
HOST : 193.44.5.2 : SYS4 : 9750 : PRIMOS : TCP/FTP, TCP/TELNET :
```

Note that the system NPRIME has both a GATEWAY and a HOST entry in the HOSTS.TXT file. NPRIME serves as a gateway and a destination host for file transfers and remote logins from SYS1 and SYS2.

LHC Directive

Each 50 Series system has one LHC. The physical device address for the LHCs in SYS1, SYS2, and SYS4 is 51₈. SYS3 has the controller on device address 52₈. Therefore, each system's system configuration file has the following entries for the LHC directive:

<i>System</i>	<i>LHC Directive</i>
SYS1	LHC 0 51
SYS2	LHC 0 51
SYS3	LHC 0 52
SYS4	LHC 0 51

THISHOST File

The SYSNAM directive in the system configuration files of SYS1, SYS2, SYS3, and SYS4 sets the names of these systems at cold start as SYS1, SYS2, SYS3, and SYS4, respectively. The names of all 50 Series systems are valid under the DoD and can use the default system name in the THISHOST file.

The THISHOST file in the TCP/IP* directory of SYS1 has the following entry:

```
;
device address = 51 : internet address = 129.42.0.1 :
```

The THISHOST file in the TCP/IP* directory of SYS2 has the following entry:

```
;
device address = 51 : internet address = 129.42.0.2 :
```

The THISHOST file in the TCP/IP* directory of SYS3 has the following entry:

```
;
device address = 52 : internet address = 193.44.5.1 :
```

The THISHOST file in the TCP/IP* directory of SYS4 has the following entry:

```
;
device address = 51 : internet address = 193.44.5.2 :
```

HOSTNAME_CONFIG File

SYS1, SYS2, SYS3, and SYS4 have the following entries in their HOSTNAME_CONFIG files:

```
CACHE AGE = 25
SAVE CACHE = FALSE
SERVER ADDRESS = 129.42.0.4
```

The server address is the Internet address of NPRIME2.

CONFIG_NTS Session

The following listing is from the CONFIG_NTS session done for Sample Network 4. It describes a configuration file that is installed on all four 50 Series systems. There are no LTSs on the network. Do not configure the other vendors' systems to look like LTSs. You must specify that unconfigured nodes are permitted. You create this listing by entering 6 (display, list or spool configuration) on the CONFIG_NTS main menu after you have created and saved the configuration. See Figure 15-8.

The COMM_CONTROLLER Command

From the supervisor terminal of each 50 Series system, issue the following COMM_CONTROLLER commands to downline load the LHCs for PRIMOS TCP/IP:

<i>System</i>	<i>Command</i>
SYS1	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 51 -PR TCP_TEL
SYS2	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 51 -PR TCP_TEL
SYS3	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 52 -PR TCP_TEL
SYS4	OK, COMM_CONTROLLER -LOAD -DEV LHC -DA 51 -PR TCP_TEL


```
CONFIG_NTS, revision 22.0                Dump of configuration file:
                                         <OURDISK>ADMIN>EXAM1_CONFIG.CONFIG
```

Sample configuration #4

File created on 16 Sep 88 at 13:29 by ADMIN

Last edited on 16 Sep 88 at 16:45 by ADMIN

- - - LAN: EXAM4 - - -

Physical LAN type: IEEE 802.3 10M 500 meter LAN

Unconfigured nodes are permitted.

Network management functions are not configured.

Host name	LHCs
-----	-----
SYS1	LHC00
SYS2	LHC00
SYS3	LHC00
SYS4	LHC00

No LTSs are configured.

Figure 15-8
Summary of Configuration File for EXAM4NET

Monitoring PRIMOS TCP/IP

This chapter describes the following features that enable a Network Administrator to monitor PRIMOS TCP/IP:

- LHC polling and recovery mechanism
- PRIMOS TCP/IP, LAN300 Network Management, and DSM files that log event messages
- Use of the LIST_LHC_STATUS command to monitor a LAN Host Controller 300 (LHC) connected to a 50 Series host running PRIMOS TCP/IP

LHC Polling and Recovery

The LHC polling and recovery mechanism provides continuous and automatic monitoring of LHCs. The LAN300 Network Management Server (NM_SERVER) is responsible for detecting failures of local LHCs and recovering from those failures without manual intervention by the Network Administrator or operator.

When an LHC recovers from failure, the following events take place automatically:

- TCP/IP_MANAGER is notified that an LHC failed and the event is reported to DSM.
- An upline dump of the memory image of the failed controller is sent to a default file (LHC_hostname-deviceaddr.date.time in the UP_LINE_DUMP*>LAN300 directory).
- The controller is tested and the file that was on the controller at the time of failure is downline loaded.
- The Network Management Server (NM_SERVER) running on the host is reconnected to the network management process running on the controller.

If an LHC upline dump is unsuccessful, the LHC can still recover from failure.

The following conditions can disable the LHC recovery mechanism:

- An LHC experiences more than three failures and recoveries per hour.
- A fatal hardware problem occurs.

- The Network Management Server stops running.
- The recovery mechanism cannot recover from a failure.

Log Files

PRIMOS TCP/IP, LAN300 Network Management, and DSM generate a number of different log files as part of the startup, shutdown, and recovery process. This section describes those files.

Note

Log file messages are diagnostic messages intended primarily for the use of your customer service representative.

PRIMOS TCP/IP Log Files

PRIMOS TCP/IP startup and shutdown procedures generate the following local log files, which are in the TCP/IP*>LOG_FILES directory:

FTP_SS_LOG

Records error messages from the FTP server process (TCPFTP_SERVER nn).

FTP_SP_LOG

Records error messages from the FTP server phantom process (TCPFTP_SERVER_PHANTOM nn).

FTP_UP_LOG

Records error messages from User FTP.

HOSTNAME_LOG

Records information when the Hostname Service is initialized and when the Hostname Service retrieves information from a remote host.

MANAGER_LOG

Records error messages from the PRIMOS TCP/IP managing process (TCP/IP_MANAGER).

PORTAL_LOG

Created when you issue the START_FTP_SERVER command and overwritten each time the START_FTP_SERVER command is issued.

SERVER_FTP_LOG

Created when you issue the `START_FTP_SERVER` command and overwritten each time the `START_FTP_SERVER` command is issued.

SMTP_LOG

Created when you issue the `START_SMTP` command and overwritten each time the `START_SMTP` command is issued.

START_TCP/IP_LOG

Created when you issue the `START_TCP/IP` command and overwritten each time the `START_TCP/IP` command is issued.

The `MAILER_DAEMON` logs events to sequentially numbered files named `MAIL.LOGnn` in the `MAILER*>LOG` directory.

LAN300 Network Management Log Files

LAN300 Network Management logs events to the following log files:

NETWORK_MGT*>NMSR.LOG

Records events that originate on the LHC and that are detected by the Network Management Servers running on the host and on the controller.

NETWORK_MGT*>DLL.LOG

Records downline load event messages.

NETWORK_MGT*>ULD.LOG

Records upline dump event messages.

DSM Log Files

LAN300 Network Management provides a mechanism through which event conditions that arise within the LAN300 NM software can be reported to the DSM event logging facility. Event conditions are defined as informational, warning, or alarm states that should be reported to a centralized log. A Network Administrator or operator can use the centralized log to analyze network operations. Event reports that originate within the LAN300 Network Management Server are logged directly to DSM. Event reports that originate within the LHC cannot be logged directly to DSM, but are sent to the Network Management Server, which, in turn, logs events to DSM.

When you configure your system with the DSM utility `CONFIG_UM` to receive unsolicited messages from PRIMOS TCP/IP, you can log these messages to `TCP/IP*>DSM_LOGFILE`. If you do not explicitly define a log file, DSM logs messages to a default file named `DSM*>LOGS>UMH>DEFAULT.LOG` file.

The following messages are examples of PRIMOS TCP/IP messages logged to DSM:

```
Starting up PRIMOS TCP/IP
PRIMOS TCP/IP startup was successful
```

```
Starting up PRIMOS TCP/IP Server FTP for device 56
PRIMOS TCP/IP Server FTP startup was successful for device 56
```

```
Shutting down PRIMOS TCP/IP
PRIMOS TCP/IP shutdown was successful
```

```
PRIMOS TCP/IP has entered a recovery state
PRIMOS TCP/IP recovery was successful
```

LIST_LHC_STATUS Command

LIST_LHC_STATUS is a PRIMOS command that enables Network Administrators and operators to request and examine information pertaining to an LHC running PRIMOS TCP/IP. You can use the online operational status information retrieved directly from an LHC to determine current utilization and general status of the LHC located on the 50 Series system.

The LIST_LHC_STATUS command presents information relevant to PRIMOS TCP/IP in an overview screen, which displays identification and performance data.

Command Format

The LIST_LHC_STATUS command has the following format for PRIMOS TCP/IP:

```
LIST_LHC_STATUS -DEST_LHC_NUMBER number -PERFORMANCE -ALL
                  -DLHC                               -PERF
```

LIST_LHC_STATUS command options have the following meanings:

-DLHC *number*

The logical controller number that identifies the LHC on the local node. *number* is an octal number that is in the range 0 through 7, inclusive. It is the number that you entered in the CONFIG file with the LHC directive.

-PERF

Selects the Performance screen. The Performance screen provides a single screen that describes the operational state of the controller. The next section describes the fields on the Performance screen.

Performance Screen

The following example of the LIST_LHC_STATUS command displays the Performance screen, as shown in Figure 16-1.

OK, **LIST_LHC_STATUS -DLHC 2 -PERF**

[LIST_LHC_STATUS Rev. 22.0 Copyright (c) 1988, Prime Computer, Inc.]

```

Host Name: HOST2          LHC Number: 02   Address: 08-00-2F-00--01-02
Hw/Fw Rev: 08.00/02.14   Slot Number: 13  Device Address: 32

State      : OPERATIONAL      Active Protocols: TCP/TELNET
LAN Name   : LAN1
Load File  : TCP.DL

Performance Statistics                                     Count
-----
Active connections                                     : 5
Frames transmitted                                       : 8014
Frames transmitted with error                           : 6
Frames retransmitted                                    : 6
Frames received                                         : 11118
Frames received with network (CRC/Alignment) error      : 0
Frames lost due to internal (resource) error            : 1
Percent CPU idle time                                   : 89.3%
Current percent data buffer availability                : 96.4%
Highest percent data buffer availability                : 95.8%
Alarms reported                                         : 0

```

Figure 16-1
Performance Screen

The fields in the Performance screen have the following meanings:

Host Name

The name of the local 50 Series host.

LHC Number

The logical controller number of the LHC on the local node that you specified with the -DLHC option of the LIST_LHC_STATUS command.

Address

The 48-bit LAN300 address of the the LHC.

Hw/Fw Rev

Hardware and firmware revision levels of the LHC.

Slot Number

The slot number of the LHC in the backplane of the 50 Series host.

Device Address

The device address of the LHC entered in the CONFIG file and the THISHOST file.

State

The current condition of the LHC. OPERATIONAL is the only state displayed at this time.

Active Protocols

The protocol token combination downline loaded on to the LHC.

LAN Name

The name of the LAN300 to which the LHC is attached.

Load File

The filename of the LHC downline load file.

Active connections

The number of link layer connections currently active on the LHC.

Frames transmitted

The number of link layer frames transmitted by the LHC.

Frames transmitted with error

The number of link layer frames transmitted by the LHC with error. A frame transmitted with error implies that while sending a frame, the LHC detected a physical network error such as carrier loss or too many collisions and was forced to abort the transmission.

Frames retransmitted

The number of link layer frames retransmitted by the LHC. Link layer protocol was retransmitted because the destination did not receive those frames.

Frames received

The number of link layer frames received by the LHC.

Frames received with network CRC/Alignment error

The number of link layer frames received by the LHC with CRC or alignment errors. CRC and alignment errors are network error conditions that the LHC can detect.

Frames lost due to internal (resource) error

The number of link layer frames dropped by the LHC because it was unable to allocate the resources, such as buffer space necessary to receive frames.

Percent CPU idle time

A percentage value indicating the amount of time that the LHC CPU is idle, that is, not occupied processing network activity.

Current percent data buffer availability

A percentage value indicating the amount of data buffer space currently available for use on the LHC.

Lowest percent data buffer availability

A percentage value indicating the lowest amount of data buffer space available on the LHC since it began operation.

Alarms reported

The number of alarms reported by the LHC.

User TELNET Error Messages

This chapter describes the error messages that the user TELNET program can display.

The following messages can be returned to a local TELNET user on a 50 Series system:

An internal error has occurred

User TELNET has experienced an internal software error. Report this to your System Administrator.

Cannot allocate memory for the TELNET message text file

User TELNET has experienced an internal software error. Report this to your System Administrator.

Cannot initiate a connection to the TCP/IP controller

User TELNET cannot open a connection to the remote system because the LHC300 controller on the local host is not in operation.

Cannot open TELNET message text file

User TELNET has experienced an internal software error. Report this to your System Administrator.

Cannot read TELNET message text file

User TELNET has experienced an internal software error. Report this to your System Administrator.

Command syntax error

You entered an invalid user TELNET command. Refer to Chapter 2, User TELNET, for the supported TELNET commands and the correct syntax.

Connection closed by remote machine

The remote system broke the connection. Attempt to establish a new connection.

Connection to remote machine not yet established. Please wait for prompt from remote machine

You must be connected to the remote machine before you enter a TELNET command.

Erroneous option negotiation command: *command*

You switched on option negotiation. User TELNET received an invalid *command* from the remote host.

Host name to Internet Address mapping cannot be performed

The system name or Internet address that you entered is not in the HOSTS.TXT file or could not be retrieved by the Hostname Service. Check that the name or address that you entered is correct.

An internal error has occurred

User TELNET has encountered an internal software error. If the error occurs again, contact your System Administrator.

Invalid command

You entered an invalid command. Refer to Chapter 2, User TELNET, for a description of valid commands.

Multiple arguments should be quoted

You invoked the ! command to enter a PRIMOS command longer than one word but you did not surround the PRIMOS command with quotation marks.

Option negotiation option is out of range: *integer*

You switched on Option negotiation. User TELNET received a value that is out of range. *integer* is the value.

Remote not responding. Still trying...

TELNET is attempting to establish a connection to a remote host. You can wait, enter the escape character, or enter Ctrl P. If you enter the escape character, TELNET attempts to return you to Command mode. If you enter Ctrl P, TELNET gives you the option of exiting TELNET to PRIMOS command level, terminating the current TELNET session, or continuing the attempt to establish a connection. The message may be repeated more than once.

The escape character you entered is not valid

You entered an invalid escape character. Refer to the list of valid escape characters in Chapter 2, User TELNET.

The TCP/IP controller has gone down

The LHC300 controller on the local host is not in operation.

The TCP/IP product has not been started

You cannot invoke user TELNET unless PRIMOS TCP/IP has been started. Contact your System Administrator to start TCP/IP.

Unknown error trying to initialize the TELNET message text file

User TELNET has experienced an internal software error. Report this to your System Administrator.

You are only allowed one connection at a time

You cannot open more than one TELNET connection simultaneously. Close your current connection before you attempt to open another connection.

Note

TELNET can also return the socket library error messages described in Appendix G.

PRIMOS FTP Program Messages

This appendix lists and describes the program messages that PRIMOS FTP returns to local and remote users, respectively.

User FTP Program Messages

PRIMOS FTP may return the following messages to local 50 Series users. These messages are not numbered:

Abnormal condition detected by controller

An abnormal condition (socket error) occurred while connected to a remote host.

Already connected to *remote_host*. Close existing connection first

You must close a connection to one remote system with the CLOSE command before you can open a connection to another remote system with the OPEN command.

Ambiguous command

You entered an abbreviation for an FTP command. The abbreviation is not unique and can refer to more than one FTP command.

ASCII type could not be set for directory listing transfer

The remote host did not accept the TYPE A command that PRIMOS FTP sent. If the current transfer type is binary, FTP tried to set the type to ASCII temporarily before it transferred any listing.

Can't change to local *directory_name*

PRIMOS FTP cannot attach to the directory on the local system. Check whether the pathname that you specified is correct or whether that directory exists.

Can't create *pathname*; *detailed_err_msg*

FTP cannot create a temporary file or the file that you specified in a pathname. *detailed_err_msg* is a detailed error message that describes the problem.

Can't create temporary list of files to be sent; *detailed_err_msg*

You do not have sufficient access rights to create a temporary file for the list of entries you selected with the MPUT command. *detailed_err_msg* is a detailed error message that describes the problem. When you invoke the MPUT command, FTP creates a temporary file with a unique name in your local directory. FTP then attaches to your local directory and reads through the listing of files. To invoke MPUT, you need ADLU access rights on your local directory, R access rights for the temporary file created and files to be sent, and U access to all parent directories.

Can't establish data connection

PRIMOS FTP on the 50 Series system cannot open a connection to the remote FTP server. No data can be transferred. This message may mean that your system is not connected to the network, or the remote system is powered off, or the remote machine is in single-user mode.

Can't read the list of files to be sent; *detailed_err_msg*.

You invoked either the MGET or MPUT command. FTP cannot open the temporary file that contains the generated list of directory entries to be retrieved (MGET) or sent (MPUT). You do not have sufficient access rights or the file is in use. *detailed_err_msg* is a detailed error message that describes the problem. To invoke MGET, you need ADLU rights on your local directory, R access for the temporary file created and the files to be retrieved, and U access for all parent directories. (You also need access rights to the directory on the local system.) To invoke MPUT, you need ADLU rights on your local directory, R access for the temporary file created and files to be sent, and U access for all parent directories.

Can't reattach to original directory. Current local directory is *pathname*.

When you invoked the MPUT command, you specified a directory in *pathname* that is not in your current directory. After FTP generated the list of entries you selected, FTP could not return to your original current directory. Your access rights to your directory may have been changed after you entered the MPUT command.

Cannot initialize connection to controller

You must be a member of the .UBI\$ ACL group to perform the requested operation.

Cannot initiate data connection

The PORT command failed or it could not setup a socket to listen and to accept the incoming data connection.

Cannot open connections to any remote host. The TCP/IP product is not running on your local system. Contact your system administrator to get TCP/IP started

You attempted to open a connection but PRIMOS TCP/IP is not running on the local system. After these messages are displayed, FTP returns you to command level.

Cannot specify local port for data connection to remote host

You requested a data transfer that requires a new port to be specified to the remote server. However the PORT command failed.

Can't open *pathname*; *detailed_err_msg*

PRIMOS FTP cannot open the file on the 50 Series system that you want to transfer to a remote system. *detailed_err_msg* is a detailed error message that describes the problem. Check whether you have access rights to that file. Also check whether the file exists or, if it exists, whether the file is in use.

Command argument too long

The argument that you entered is too long. You probably forgot to enclose a multiple argument with a single or double quotation mark.

Command error

You provided arguments for a command that does not require any arguments.

Controller down

You cannot open a connect to the remote host because the LHC controller on the local host is not in operation.

Could not generate the list of files to be retrieved from the remote host.

You invoked the MGET command. An error occurred while FTP attempted to select the remote directory entries (for MGET) that matched the wildcard specification.

Could not generate the list of files to be sent to the remote host

You invoked the MPUT command. An error occurred while FTP attempted to select the local directory entries (for MPUT) that matched the wildcard specification.

Could not mark the file as originally of an odd/even byte size

After FTP transfers a binary file, it attempts to mark the file as having an odd or even number of bytes. If FTP cannot mark the file, it displays this message.

Could not reattach to original directory. Current local directory is *pathname*

For FTP to mark or verify that a file has an odd or an even number of bytes, FTP must be attached to the directory in which the file resides. If, after FTP attaches to the new directory, the access rights on the original directory are modified, and FTP cannot reattach to the original directory, this message is displayed. *pathname* is the name of your current directory.

Could not rename temporary file. Retrieved data saved in *pathname*

If you specify an existing file when you retrieve a remote file and you want to overwrite it, FTP creates a temporary file. Once the transfer is successful, the existing file is deleted and the temporary file is renamed to the specified file. If the existing file cannot be deleted or the temporary file cannot be renamed, FTP saves the file in *pathname*.

Could not verify if the file was originally of an odd\even byte size

Before FTP transfers a binary file, it checks to see whether the file has an even number or an odd number of bytes. If the file has an an odd number of bytes, FTP strips off the extra null character at the end of the file before it transfers it to the remote system. If FTP cannot read the file's attributes, it displays this message and assumes the file has an even number of bytes.

Data connection failure: controller interface error

The data connection failed in the middle of a file transfer. The local or remote host broke the connection.

Data transfer aborted

You entered , the remote host went down during a data transfer, or the data connection has been disconnected.

Disconnected from remote host

The remote host, the remote server, or the network broke the connection.

Error in closing connection

An error occurred while PRIMOS FTP was trying to close the connection to the remote system. The error does not prevent you from entering another command. If the error message is repeated, contact your Network or System Administrator.

Error in host-controller interface

An internal software error related to sockets has occurred.

Flushing I/O buffer failed; *detailed_err_msg*

While FTP was retrieving a file from the remote host, it encountered an error writing to the disk on the local host. *detailed_err_msg* describes the error encountered.

Hardware error: cannot allocate storage space

The disk is full on the local system. FTP cannot complete the requested operation.

Insufficient access rights or directory does not exist

You have insufficient access rights to perform the operation that you want. One of the parent directories does not exist. Check whether the directory exists on the local 50 Series system.

Refer to Chapter 3, Using PRIMOS FTP on a 50 Series System, for a description of the access rights that you require.

Insufficient access rights to connect to controller

You must be a member of the .UBI\$ ACL group to perform the requested operation.

Insufficient system resources to perform requested operation.
Exceeded the limit of open connections per user session. Allow some time for the local system to close lingering connections. Retry operation later

All available connections are in use or have not been released by the LHC controller. FTP cannot create a socket for the control connection.

Insufficient system resources to perform requested operation.
Exceeded the limit of open connections per user session. Allow some time for the local system to close data connections. Retry operation later

All available connections are in use or the LHC controller has not released all available connections. Consequently, FTP cannot create a socket for the data connection.

Insufficient system resources to perform requested operation. Unable to connect to controller. All available connections are in use

Sufficient system resources are not available to perform the requested operation. The number of FTP users or TELNET users or both has exceeded the limit on the local system.

Internet address specified for local host is not available. Verify the internet address specified in THISHOST file

The local host is inaccurately specified in the THISHOST file.

Invalid reply from the FTP server on the remote host

The remote host sent a reply that is not preceded by a code.

Login failed

You entered an invalid user ID or password.

MGET command aborted. Control connection failure.

The control connection was broken while FTP was retrieving multiple files. The remote host went down, network reset, local host went down, or you entered ☐ C ☐ M ☐ P.

MPUT command aborted. Control connection failure.

The control connection was broken while FTP was sending multiple files. The remote host went down, network reset, local host went down, or you typed ☐ C ☐ M ☐ P.

Multiple arguments should be quoted

Surround the argument that you entered with the ! command with single or double quotation marks.

Network reset

You cannot open a connection to the specified host or complete your transaction because the network reset the connection.

Network down

The network is down or cannot complete the operation that you requested. Try again later. Also, check to see whether your system is physically connected to the network.

Network unreachable

The network of the host that you specified cannot be reached from your network. You may have entered an incorrect Internet address with the FTP command or the OPEN command.

No connection open

You invoked the CLOSE command but no connection has been opened.

No entries selected

You used the MGET or MPUT command with a wildcard name. FTP searched the local or remote directory but did not find a filename or pathname that matched the wildcard name.

Partially retrieved data saved in *pathname*

The file transfer was aborted in mid-stream but FTP retrieved some data from the remote host. The data are stored in *pathname*.

pathname already exists, do you wish to overwrite it (y/n)?

You specified a local filename as an argument with the RETR, RECV, or GET command. The filename that you specified already exists.

pathname (dir) already exists, do you wish to overwrite it (y/n)?

You specified a local directory name as an argument with the RETR, RECV, or GET command. The name that you specified already exists.

pathname does not exist

The file that you want to send to the remote host does not exist.

pathname is not a valid file. Operation illegal on a *file_type*

You specified a *file_type* with the SEND command that does not denote either a SAM or DAM file. You must specify a filename.

Pathname already exists. Operation illegal

You specified a pathname with the GET command that already exists on the local host.

Read failed; *detailed_err_msg*

PRIMOS FTP cannot read the file that you specified. *detailed_err_msg* describes the problem.

Remote host not connected, open using OPEN command

You invoked a command which requires that you have a connection to a remote host. Establish that connection. Use the OPEN command.

Remote host unreachable

The remote server is not running or the network is down.

RENAME failed in RNFR command

You requested that FTP rename a non-existent file on the remote host or the RNFR command failed on the remote host.

Software error; *detailed_err_msg*

FTP has detected an error in the TCP/IP code. Report this error to your System Administrator. *detailed_err_msg* describes the error.

TCP connection refused

FTP cannot connect you to the specified host. The remote host or the FTP server on the remote host is not running.

TCP connection reset

FTP cannot connect you to the specified host.

TCP connection timed out. Still trying...

An internal event has disrupted the data connection. FTP is attempting to reestablish the data connection and complete the operation. FTP may have transferred data. You are still connected to the remote system. This message may be repeated several times in succession. Wait until FTP completes the operation or use to abort the session.

TCP connection timed out. Unable to connect to remote host.

You cannot connect to the specified host or complete your transaction because the time allotted for establishing the connection has expired. The TCP/IP program on the remote host is probably not running.

Trying MKD command, please wait

PRIMOS FTP transmitted a command (XMKD) that the remote host does not recognize.

Trying PWD command, please wait

PRIMOS FTP transmitted a command (XPWD) that the remote host does not recognize.

Trying RMD command, please wait

PRIMOS FTP transmitted a command (XRMD) that the remote host does not recognize.

Trying XCWD command, please wait

PRIMOS FTP transmitted a command (XCWD) that the remote host does not recognize.

Unable to connect to remote host

PRIMOS FTP cannot connect to the remote system to complete the operation that you specified. Check whether the system name or the Internet address is correct or whether the system is listed as a valid destination in the PRIMOS TCP/IP configuration file (HOSTS.TXT).

Unimplemented mode/type/structure combination

You specified an option for either the MODE, TYPE, or STRUCTURE command that FTP does not support. Refer to Chapter 4, A PRIMOS FTP Commands Reference, for the options that each of these commands supports.

Unknown host name

PRIMOS FTP does not recognize as valid the system name that you specified. Check whether the host name that you entered is correct or whether the system is listed as a valid destination in the TCP/IP configuration file (HOSTS.TXT).

Unknown system error. ICE of user's environment recommended. If problem persists, contact your system administrator.

A system error occurred while you were connected to a remote host. Use the ICE command to reinitialize your environment.

Unrecognized command. *command*

You entered a command that PRIMOS FTP does not support or recognize. Refer to Chapter 4, A PRIMOS FTP Commands Reference, for a list of valid commands.

User has insufficient access rights to *pathname*

You do not have the correct access rights to use the file for listing, retrieving, or sending the file.

Write failed; *detailed_err_msg*

PRIMOS FTP could not write to the file on the 50 Series system. Check whether the filename is correct or whether you have the proper access rights to that file. *detailed_err_msg* describes why the write operation failed.

Wrong number of arguments

You entered too many or too few arguments with a PRIMOS FTP command. Check the correct format of the command in Chapter 4, A PRIMOS FTP Commands Reference.

FTP Server Process Program Messages

Messages that the FTP server process generates are numbered. These messages are returned to a remote user who is sending a file to or retrieving a file from a 50 Series system. The messages are listed according to their numeric reply codes.

Reply Codes

The first digit of the reply codes indicates their meaning as described in the following list:

<i>Number</i>	<i>Meaning</i>
1nn	A preliminary reply that indicates another message will be displayed. This is a reply to requests for information. You may not issue another command until a second message is displayed.
2nn	Your first command was successful. You can issue another command.
3nn	PRIMOS FTP or the remote FTP server needs more information before your command is processed.
4nn	Your command is valid but cannot be processed because either the network or the remote system is down. The error is not fatal. You can enter the command again.
5nn	You specified either the wrong command or an incorrect option to a command.

The second digit of the reply codes indicates their meaning as described in the following list:

<i>Number</i>	<i>Meaning</i>
n0n	Indicates a syntax error
n1n	Indicates a reply for information
n2n	Refers to the status of a data connection
n3n	Refers to the status of a login request
n4n	Undefined
n5n	Refers to the status of the file system

Messages

This section describes PRIMOS FTP server messages numbered above 300. The following error messages can be displayed when a user on a remote system attempts to transfer a file to or retrieve a file from the PRIMOS FTP server on a 50 Series system:

331 Enter PASS Command

You must enter a password to connect to the 50 Series system.

332 Need project id for login; use ACCOUNT command.

You may be required to enter a project ID to log in. If the System Administrator of the 50 Series system did not specify a default project and you entered , your attempt to log in failed. If you could not log in after you entered a project ID or after you entered , contact the System Administrator of the 50 Series system.

350 RNFR accepted, please send RNTD next

You entered the rename from (RNFR) command. Now you must enter the rename to (RNTD) command to change the name of the file.

425 Can't establish data connection

The data connection between your system and the remote 50 Series system cannot be established to perform the file transfer. The network or the remote system may be down.

426 Data connection (host interface) failure; *detailed_err_msg*

The data connection between your system and the remote 50 Series system cannot be established to perform the file transfer. *detailed_err_msg* describes the reason for the failure.

430 Login attempts exceeded

You have exceeded the number of times that you are allowed to attempt to log in. Check whether your user ID or password or both are valid.

450 Can't open *filename*; *detailed_err_msg*

PRIMOS FTP server cannot open the file that you specified. Check whether the file exists, whether you or the PRIMOS FTP server have access to the file, or whether the file is in use. *detailed_err_msg* is a detailed error message that describes the problem.

450 *filename* exists as non-directory

The *filename* that you specified exists as a directory on the 50 Series system.

450 Can't create *pathname*; *detailed_err_msg*

FTP cannot create the file specified in *pathname*. Check whether the *pathname* that you specified is correct or whether you or the PRIMOS FTP server have the proper access rights to perform this operation. *detailed_err_msg* is a detailed error message that describes the problem.

451 Can't start list command; *detailed_err_msg*

The PRIMOS FTP server cannot list the files in the remote 50 Series directory. *detailed_err_msg* is a detailed error message that describes the problem. Check whether the pathname that you specified is correct or whether you or the PRIMOS FTP server have the proper access rights to perform this operation.

451 Could not remove *directory_name*; The directory is not empty.

The PRIMOS FTP server could not delete the directory that you specified. Check whether the pathname that you specified is correct, whether the directory does exist, or whether you or the PRIMOS FTP server have the proper access rights to perform this operation.

451 Unable to create *directory_name*

The PRIMOS FTP server could not create the directory that you specified. Check whether the pathname that you specified is correct, or whether you or the PRIMOS FTP server have the proper access rights to perform this operation.

500 Bad password condition encountered

You entered an invalid password. Enter the correct password.

500 Command *command_name* not recognized

The PRIMOS FTP server does not recognize the command that you entered. Refer to Chapter 5, Accessing PRIMOS FTP From a Remote System, for a list of commands that PRIMOS FTP supports.

501 *address* is not a valid host/port specification

The host or port *address* that was sent is incorrect. If you entered this information with the PORT command, check whether the address is valid. If your system is sending this information and the error message is repeated, contact your System or Network Administrator.

501 Syntax error

You entered a command with syntax that the PRIMOS FTP Server does not recognize. Enter the command again with the correct syntax.

501 *port-address/host-address* argument required

The PORT command requires a valid *port-address* or a valid *host-address*. Enter the PORT command again with the correct information.

503 Haven't received a RNFR

You entered an RNTD command (or its equivalent) but did not enter an RNFR command (or its equivalent). The RNTD command must be preceded by the RNFR command.

503 Password is not expected now

You entered a password when it is not required. If a password is required, you will be prompted to enter one.

503 Project/Account is not expected now

You entered either a project ID or an account number when it was not required. If either a project ID or account number is required, you will be prompted to enter one.

503 Supply username and password before project

Enter your user ID and password before you enter your project ID.

504 Command not implemented for that parameter

You entered a command with an invalid argument. Refer to Chapter 4, A PRIMOS FTP Commands Reference, for a list of valid arguments that the command supports.

504 Mode not implemented

You entered an argument with the MODE command that PRIMOS FTP does not support. At Release 2.0, PRIMOS FTP supports STREAM mode, which is the default.

504 Unimplemented structure type

You specified a structure that PRIMOS FTP does not support. At Release 2.0, PRIMOS FTP supports file structure, which is the default.

504 Unimplemented type

You specified a structure that PRIMOS FTP does not support. At Release 2.0, PRIMOS FTP supports Image or ASCII type, which is the default.

504 Unknown mode

You entered an argument with the MODE command that PRIMOS FTP does not support. At Release 2.0, PRIMOS FTP supports STREAM mode, which is the default.

504 User command unnecessary

The remote system does not require a user ID.

530 Access not allowed for guest users

You must log in before you enter the command that caused this error message. Use the LOG or USER command to log in.

530 Insufficient access for the user or file/directory not found

You or the PRIMOS FTP server do not have the correct access rights to perform the operation that you specified. If you enter a project ID and this message appears, contact the

System Administrator of the 50 Series system. A project may have been created after TCP/IP was installed and the ACLs on the System Administration Directory (SAD) must be reset. Refer to Chapter 5, Accessing PRIMOS FTP From a Remote System, for a description of the access rights that you and the PRIMOS FTP server need.

530 Log in with USER and PASS

To log in, use the USER command and PASS or equivalent commands. Refer to Chapter 5, Accessing PRIMOS FTP From a Remote System, for a description of these commands.

530 Login attempt failed, try again

You entered an incorrect user ID or password when you logged in. Issue the USER command and PASS command (or the equivalent on your system) and enter the correct user ID or password or both.

530 Please enter your password first

You must enter your password before you can enter any other command.

550 Can't change to *directory_name*

You cannot attach to the directory on the 50 Series system. Check whether you specified the correct pathname. You may not have sufficient access rights to the directory that you specified. Check with the System Administrator of the 50 Series system to determine whether you or the PRIMOS FTP server have access to the directory.

550 Can't change to parent directory

You cannot attach to the top-level directory on the 50 Series system. Check whether you specified the correct pathname. You may not have sufficient access rights to the directory that you specified. Check with the System Administrator of the 50 Series system to determine whether you or the PRIMOS FTP server have access to the directory.

550 Can't delete *directory_name* or *filename*; *detailed_err_msg*

You cannot delete the directory or file that you specified on the 50 Series system. *detailed_err_msg* is a detailed error message that describes the problem. Check whether you specified the correct pathname. You or the PRIMOS FTP server may not have sufficient access rights to the directory or file. Check with the System Administrator of the 50 Series system to determine whether you or the PRIMOS FTP server have access to the directory or file that you want to delete.

550 Can't find *filename*; *detailed_err_msg*

FTP cannot find the filename that you specified. *detailed_err_msg* is a detailed error message that describes the problem. Check whether you specified the correct name. You or the PRIMOS FTP server may not have sufficient access rights to the directory. Check with the System Administrator of the 50 Series system to determine whether you or the PRIMOS FTP server have access to the directory.

550 Can't find *directory_name*; *detailed_err_msg*

You cannot attach to the directory on the 50 Series system. *detailed_err_msg* is a detailed error message that describes the problem. Check whether you specified the correct pathname. You or the PRIMOS FTP server may not have sufficient access rights to the directory. Check with the System Administrator of the 50 Series system to determine whether you or the PRIMOS FTP server have access to the directory.

550 Can't open *directory_name*; *detailed_err_msg*

You cannot attach to the directory on the 50 Series system. *detailed_err_msg* is a detailed error message that describes the problem. Check whether you specified the correct pathname. You or the PRIMOS FTP server may not have sufficient access rights to the directory. Check with the System Administrator of the 50 Series system to determine whether you or the PRIMOS FTP server have access to the directory.

550 PRIMOS file error - aborting *PRIMOS_message*

PRIMOS encountered a problem while performing the operation on the file. *PRIMOS_message* is the message returned by PRIMOS.

550 Unable to obtain user default project ID

You entered in response to the ACCOUNT command prompt. You must enter a valid project ID to log in or your System Administrator must assign you to a default project before you can log in.

551 Can't change to *directory_name*

You cannot attach to the directory on the 50 Series system. Check whether you specified the correct pathname. You may not have sufficient access rights to the directory. Check with the System Administrator of the 50 Series system to determine whether you or the PRIMOS FTP server have access to the directory.

552 Flushing I/O buffer failed; *detailed_err_msg*

The PRIMOS software interface to the LHC300 detected an error. *detailed_err_msg* is a detailed error message that describes the problem. If *detailed_err_msg* is displayed again, report it to your Network or System Administrator.

552 Write failed; *detailed_err_msg*

The PRIMOS software interface to the LHC300 detected an error. If the error message is displayed again, report it to your Network or System Administrator.

554 Unimplemented mode/type/structure combination

You specified a combination of mode, type, or structure that PRIMOS FTP does not support. At Release 2.0, PRIMOS FTP supports STREAM mode, ASCII or Image type, and FILE structure.

MAIL Messages

This appendix lists and describes the error and program messages that PRIMOS TCP/IP MAIL can return to a user on a 50 Series system.

Attempting to find your login directory

MAIL locates your login directory when you issue the MAIL command to initialize your mailbox.

Caller not authenticated

A MAIL message was accepted from an unconfigured or incompletely configured host. The MAIL message may be from an unreliable source. The MAIL header and message may contain invalid information. The improperly configured host generates this message, which is displayed in the MAIL message header on the Received: line.

Cannot accept; domain not known

You entered an incorrect MAIL address. A host name that contains a period is not valid.

Cannot write to mailbox; user not known

MAIL cannot match the user ID that you specified.

Can't access mailbox file

MAIL cannot access your mailbox to read your messages. Your mailbox is a subdirectory named with your user ID under the MAILER*>MBOX directory.

Can't access *message*

MAIL cannot append *message* to the file that you specified. Check that the file exists or that you have access to the file.

Couldn't open *filename* for read access

MAIL cannot read the file that you want to send. Check that the file exists or that you have access to the file. You may have specified the mail ID of the recipient as the *filename*.

Couldn't open temporary file

MAIL attempted to copy a file that you specified and send it to the recipient. The attempt to copy the file failed.

Could not delete *messages*

MAIL cannot delete the messages that you marked for deletion.

Could not create *filename*

MAIL cannot create *filename* that you specified with the FILE or the APPEND command. Check your access rights to the directory in which you issued the command.

Could not forward for *n* days

MAIL cannot reach the recipient on the destination system. One of the hosts along the mail route has been inaccessible for at least *n* days.

Did not create *filename*

You specified `-ALL` as *filename* with the FILE or APPEND command. `-ALL` is not a valid filename.

Error attempting to set forwarding, not set

MAIL encountered an error while attempting to execute the MAIL `-SET_FORWARD_TO` command.

Error code 327 from routine `crepw$`

MAIL attempted to initialize your mailbox when you entered the MAIL command. However, your system does allow password directories and this prevented the creation of your mailbox. Without a mailbox, you can send mail but you cannot receive mail. Your System Administrator must create a mailbox for you.

Failed to send to: *mail_id*

You attempted to send a message to *mail_id*. The recipient did not receive the message. The MAIL ID of the recipient is incorrect or the remote system or the network is down.

Forwarding not in effect

You entered the MAIL `-CANCEL_FORWARDING` command to cancel the directive that is issued with the MAIL `-SET_FORWARD_TO` command. However, MAIL was not set to forward messages to another address.

GUEST is not permitted to send mail

A user with the user ID GUEST cannot send mail.

mail_id does not have a mailbox

You attempted to delete your mailbox but it does not exist.

Mailbox is not empty

You cannot delete your mailbox unless you delete all the messages that it contains.

MAILER* probably does not exist

MAIL attempted to initialize your mailbox but cannot find the MAILER* directory.

Network error: code is *number*

A network error occurred while MAIL attempted to send your message. *number* describes the error.

No route known for *host.domain*

You attempted to send MAIL to a system that cannot be reached through your mail network. *host.domain* is the name of the host that cannot be reached.

Protocol error talking to SMTP server

An internal software error occurred when MAIL attempted to send your message.

Return-path too long; too many hops

MAIL cannot reach the recipient on the destination system. This message usually indicates that a routing loop has inadvertently been created in the mail network. Two or more systems are passing mail back and forth to each other. Bring this message to the attention of your Network Administrator.

Send failed

MAIL was unable to deliver your message to the designated recipient or recipients.

Unable to reset ACL on mailbox

MAIL encountered a problem when it attempted to delete your mailbox in the MAILER*>MBOX directory. Check whether your mailbox has been deleted or whether the correct ACLs are set on the mailbox if it still exists.

PRIMOS TCP/IP Configuration and Installation Error Messages

This appendix lists and describes TCP/IP error messages that can be displayed when you configure and install TCP/IP on a 50 Series system.

TCP/IPHTU Error Messages

This section describes error messages that can appear when you run the PRIMOS TCP/IP Host Table Utility (TCP/IPHTU).

THISHOST file does not exist

TCP/IPHTU could not find THISHOST in the current directory.

Premature end of file in THISHOST file

THISHOST file ended without specifying all needed information.

Device address out of range in THISHOST file

Maximum device address is 77 (octal).

No controller line entries in THISHOST file

You must specify at least one controller.

Illegal line format in THISHOST file

The THISHOST file is formatted incorrectly. Check to see whether you included all required fields and colons.

Internal error in reading THISHOST file

An internal software error occurred while TCP/IPHTU was reading THISHOST.

Host file line too long, terminating

A line in the host file is longer than 240 characters.

End of file while reading host entry, terminating
The host file ended in the middle of a HOST entry.

End of file while reading network entry, terminating
The host file ended in the middle of a NET entry.

End of file while reading gateway entry, terminating
The host file ended in the middle of a GATEWAY entry.

Cannot open hosts file for writing
TCP/IPHTU cannot open the output file to which it writes host information. Check access rights to the current directory. You should be attached to TCP/IP*.

Cannot open gateways file for writing
TCP/IPHTU cannot open the output file to which it writes gateway information. Check access rights to the current directory. You should be attached to TCP/IP*.

Illegal format, first word is not NET, HOST, or GATEWAY
Every line must either be a comment or begin with NET, HOST, or GATEWAY.

Illegal hostname specified in THISHOST file or system default
The name specified in THISHOST or the system default (if used), does not follow the Internet host name rules.

Illegal subnet mask format in THISHOST file
The subnet mask must consist of eight hex digits.

COMM_CONTROLLER Command Error Messages

This section describes the most common TCP/IP-related error messages that can be generated when a COMM_CONTROLLER command is rejected. Normally, all error messages are also logged by DSM. For a list of COMM_CONTROLLER command error messages that apply to ICS controllers and LTSSs, see Chapter 2 of the *System Administrator's Guide, Volume II: Communication Lines and Controllers*.

Error: I/O bus initialization failure.

You must issue the COMM_CONTROLLER command from a supervisor terminal. If you issue the COMM_CONTROLLER command from a supervisor terminal and this message is displayed, you may have the wrong version of the PRIMOS TCP/IP downline load file.

Error: DEVICE *type* is invalid.

The controller *type* specified must be one of the following: ICS1, ICS2, ICS3, LHC, or LTS.

Error: DEVICE not specified.

The DEVICE option was specified without supplying a valid controller type.

Error: DEVICE option must be specified.

The DEVICE option was omitted from the command line.

Error: DEVICE_ADDRESS *nn* is not octal.

The device address for a communication controller must be a two-digit octal number.

Error: DEVICE_ADDRESS has more than two characters.

The device address for a communication controller must be a two-digit octal number.

Error: DEVICE_ADDRESS not specified.

The DEVICE_ADDRESS option was specified without a valid address.

Error: DNN option is invalid for the *type* controller.

The DNN option is valid only for LTS controllers.

Error: INIT option invalid for the *type* controller.

The INIT option applies only to ICS controllers.

Error: PATHNAME *path* is invalid.

Pathname must be an existing PRIMOS pathname.

Error: PATHNAME not specified.

The PATHNAME option was specified without supplying a valid PRIMOS pathname.

Error: PATHNAME option can only be specified with LOAD or ULD.

The PATHNAME option was supplied with either the INIT or SHUTDOWN options.

Error: PROTOCOL separator *_* arranged incorrectly.

Valid protocol token combinations are connected by an underscore.

Error: PROTOCOL token *token* is duplicated.

A protocol token was supplied more than once in the command line.

Error: PROTOCOL token *token* is invalid.

The only valid protocols for Rev. 22.0 are J 25, NTS, TCP, TEL, ASYNC, SYNC, HDLC, BSCRJE, and BSCX25.

Error: PROTOCOL token is not specified.

The PROTOCOL option was specified without a valid token combination.

Error: UPLINE_DUMP option invalid for the *type* controller.

The UPLINE_DUMP option was specified for an ICS controller.

Error: cannot specify DEVICE_ADDRESS and ALL together.

Do not specify both the DEVICE_ADDRESS and ALL options on one command line.

Error: cannot specify PATHNAME and PROTOCOL together.

Do not specify both the PATHNAME and PROTOCOL options on one command line.

Error: cannot upline dump more than ONE device.

Do not specify more than one controller on the command line.

Error: either the DEVICE_ADDRESS or ALL option must be specified.

Either an LHC or an ICS controller was specified without further identification.

Error: invalid PATHNAME specified.

Pathname must be an existing PRIMOS pathname that is no longer than 128 characters.

Error: invalid response to a question.

The only valid answers to the Continue? prompt are Y or N.

Error: no main option specified (select LOAD, ULD, SH, INIT, or HELP).

The COMM_CONTROLLER command was issued without specifying an option.

Error: unknown option *option*.

The option specified is not supported.

Network Management and Monitoring Program Messages

This appendix lists and explains the error messages that can occur when a Network Administrator or operator uses the LIST_LHC_STATUS command.

Bad node address format.

The format that you specified for the source or destination node address is not valid. You can enter source and destination node addresses in either of the following formats:

Short form: `xx-xx-xx`

Long form: `xx-xx-xx-xx-xx-xx`

Each *x* is a hexadecimal number in the range 0 through F. For the pair *xx*, leading zeros are not necessary. When you use the short form, the standard Prime LAN300 prefix 08-00-2F is automatically added before the given address. The separator (-) is not part of the node address.

Bad node address, the first 6 hex digits must be 08-00-2F.

When you use the long form, the specified source or destination node address must begin with the standard Prime LAN300 address 08-00-2F. You can enter source and destination node addresses in either of the following formats:

Short form: `xx-xx-xx`

Long form: `xx-xx-xx-xx-xx-xx`

Each *x* is a hexadecimal number in the range 0 through F. For the pair *xx*, leading zeros are not necessary. When you use the short form, the standard Prime LAN300 prefix 08-00-2F is automatically added before the given address. The separator (-) is not part of the node address.

Cannot specify LHC number and node address options together.

You cannot specify both a source LHC number and source node address together or both a destination LHC number and destination node address together. Enter either an LHC number option or a node address option for the same node.

Cannot specify LAN name and LHC number options together.

Entering a LAN name along with a source or destination LHC number is invalid. Enter either one or the other.

Cannot specify LAN name and node address options together.

Entering a LAN name along with a source or destination node address is redundant and therefore illegal. Enter either one or the other.

Cannot use destination node name and address options together.

You cannot specify a destination node name and destination node address together. Enter either a destination node name or destination node address but not both.

Cannot use source node name and address options together.

You cannot specify a source node name and source node address together. Enter either a source node name or source node address but not both.

Either a destination name or address must be entered.

You must specify either the destination node name or destination node address.

Host name is invalid.

The source or destination HOST name that you specified is invalid. Host source and destination node names must begin with an alphabetic character and consist of a set of one to six legal characters. The set of legal characters are the letters A through Z, the digits 0 through 9, the period (.), the underscore (_), and the characters - # \$ * / &.

Insufficient access rights to invoke this command.

An unauthorized user invoked this command. Only User 1 or users that belong to the .NETWORK_MGT\$ ACL group are authorized to invoke this command.

Internal failure d.

An internal error occurred that is unknown to the command. Retry the command.

Internal system error.

A failure occurred while a PRIMOS routine was being executed.

Invalid connection type, PRIMENET or NTS are valid.

When you specify the -CONNECTION option, the only options that are valid are PRIMENET or NTS.

Invalid information received from LHC.

The packet received from the specified LHC contains data that is invalid.

Invalid information received from NME, bad packet length.

The Network Management Extension (NME) process that runs on the LHC received a packet that was the wrong length.

Invalid LAN name, the first character must be alphabetic.

The specified LAN name contains an illegal first character. A LAN name must begin with an alphabetic character and consist of a set of 1 to 16 legal characters. The set of legal characters are the letters A through Z, the digits 0 through 9, the period (.), the underscore (_), and the characters - # \$ * / &.

Invalid node name, it contains illegal characters.

The specified source or destination node name contains any number of illegal characters. LTS source and destination node names must begin with an alphabetic character and consist of a set of 1 to 16 legal characters. Host source and destination node names must begin with an alphabetic character and consist of a set of 1 to 6 legal characters. The set of legal characters are the letters A through Z, the digits 0 through 9, the period (.), the underscore (_), and the characters - # \$ * / &.

Invalid node name, the first character must be alphabetic.

The specified source or destination node name contains an illegal first character. Host source and destination node names must begin with an alphabetic character and consist of a set of one to six legal characters. The set of legal characters are the letters A through Z, the digits 0 through 9, the period (.), the underscore (_), and the characters - # \$ * / &.

Inter Server Communication operation failed, reason code: d.

A failure occurred while a PRIMOS routine was being executed.

LAN name length is invalid.

The specified LAN name must be 16 characters or less. A LAN name must begin with an alphabetic character and consist of a set of 1 to 16 legal characters. The set of legal characters are the letters A through Z, the digits 0 through 9, the period (.), the underscore (_), and the characters - # \$ * / &.

LAN name must be entered when the LAN name option is used.

A valid LAN name must be entered whenever the LAN name (-LAN_NAME) option is used. A LAN name must begin with an alphabetic character and consist of a set of 1 to 16 legal characters. The set of legal characters are the letters A through Z, the digits 0 through 9, the period (.), the underscore (_), and the characters - # \$ * / &.

LAN not configured.

The specified LAN name is not configured for any service.

LAN not configured for specified node.

Unable to reach specified node on the given LAN.

LHC number must be entered when an LHC number option is used.

A valid LHC number must be entered whenever the source or destination LHC options are used. LHC numbers must be in the range 0 through 7.

LHC number must be in the range 0 through 7.

The specified source or destination LHC number did not fall within the valid range of LHC numbers. LHC numbers must be in the range 0 through 7.

LHC number not configured.

The specified LHC number is not configured for any service.

LHC number not configured for the specified LAN.

The LHC number is not configured for the specified LAN. Find another LHC that is configured for the desired LAN.

Local host not configured.

The local host is not found in the Node Status database.

Local host not configured to reach the remote site.

The local host is not connected to the same LAN as the destination node.

The Network management server is not running.

The command is unable to continue processing because the local Network Management Server (NM_SERVER) is not running. Make sure that PRIMOS TCP/IP, PRIMENET (configured for LAN300), or NTS is currently executing before invoking a Network Management command.

No help available.

The commands help file is missing.

No response from destination node.

A timeout occurred while waiting for a response from the destination node.

No response from the local Network Management Server.

A timeout occurred while waiting for a response from the local Network Management Server (NM_SERVER).

No response from LHC.

A timeout occurred while waiting for a response from the specified LHC.

No response from LTS.

A timeout occurred while waiting for a response from the specified LTS.

No response from source node.

A timeout occurred while waiting for a response from the source node.

Node address must be entered when a node address option is used.

A valid node address must be entered whenever the source or destination node address options are used. Source and destination node addresses can be entered in either of the following formats:

Short form: *xx-xx-xx*

Long form: *xx-xx-xx-xx-xx-xx*

Each *x* is a hexadecimal number in the range 0 through F. For the pair *xx*, leading zeros are not necessary. When you use the short form, the standard Prime LAN300 prefix 08-00-2F is automatically added before the given address. The separator (-) is not part of the node address.

Node address must be in hexadecimal digits.

The specified source or destination node address must consist of hexadecimal digits. You can enter source and destination node addresses in either of the following formats:

Short form: *xx-xx-xx*

Long form: *xx-xx-xx-xx-xx-xx*

Each *x* is a hexadecimal number in the range 0 through F. For the pair *xx*, leading zeros are not necessary. When you use the short form, the standard Prime LAN300 prefix 08-00-2F is automatically added before the given address. The separator (-) is not part of the node address.

Node name length is invalid.

The specified source or destination node name must be 16 characters or less. LTS source and destination node names must begin with an alphabetic character and consist of a set of 1 to 16 legal characters. Host source and destination node names must begin with an alphabetic character and consist of a set of 1 to 6 legal characters. The set of legal characters are the letters A through Z, the digits 0 through 9, the period (.), the underscore (_), and the characters - # \$ * / &.

Node name must be entered when a node name option is used.

A valid node name must be entered whenever the source or destination node name options are used. LTS source and destination node names must begin with an alphabetic character and consist of a set of 1 to 16 legal characters. Host source and destination node names must begin with an alphabetic character and consist of a set of 1 to 6 legal characters. The set of legal characters are the letters A through Z, the digits 0 through 9, the period (.), the underscore (_), and the characters - # \$ * / &.

Node Status database operation failed, reason code: *d*.

A failure occurred during execution of a PRIMOS Node Status database routine. In order to use a Network Management command, make sure that PRIMOS TCP/IP, PRIMENET (configured for LAN300), or NTS is started.

Option(s) *bad_option(s)* not recognized by this command.

A Network Management command encountered an invalid option.

The Node Status database not initialized.

The local Node Status database is not initialized. Therefore, no Network Management commands execute properly.

The Node Status database is corrupted.

The local Node Status database is corrupt and, therefore, prevents any Network Management commands from executing properly.

Node not found.

The specified source or destination node was not found in the local database.

Path to local LHC not initialized.

No logical connection from the Network Management Server (NM_SERVER) to the specified LHC exists.

Remote host not configured.

The specified destination host is not configured in the Node Status database.

Source and destination nodes cannot be on different LANS.

You issued a command that attempts to cross LANs. That is not permitted. Both the source and destination nodes must be connected to the same LAN.

Specified node name not a host.

Some Network Management commands assume that a request is going to be sent to a host only. Therefore, when you specify a destination node name, make sure it is a host, not an LTS.

Specified node address not an LHC.

Some Network Management commands assume that a request is going to be sent to an LHC only. Therefore, when you specify a node address, it must be a node address of an LHC.

Synchronizer operation failed, reason code: *d*.

A failure occurred during execution of a PRIMOS synchronizer routine.

NTS-related Configuration Directives

This appendix describes any special considerations that apply to NTS-related CONFIG directives that you specify when you install NTS for PRIMOS TCP/IP (TELNET). For more information on these directives, refer to Chapter 2, Installing NTS, of the *NTS Planning and Configuration Guide*. The directives are listed in alphabetical order.

<i>Directive</i>	<i>Value for PRIMOS TCP/IP</i>
AMLCLK	Specifies the baud rate for the software programmable clock for both locally attached and NTS-attached asynchronous lines. No special considerations apply to PRIMOS TCP/IP.
AMLTIM	Sets time intervals for three variable event timers: ticks, disctime, and gracetime. No special considerations apply to PRIMOS TCP/IP.
ASYNC JUMPER	Defines the line speeds available for assigned lines connected to an LTS over the NTS network. The ASYNC JUMPER directive does not apply to PRIMOS TCP/IP because PRIMOS TCP/IP does not support assigned lines.
DISLOG	Enables or disables automatic logout when a locally attached line is disconnected. The DISLOG directive does not apply to either NTS or PRIMOS TCP/IP.
DTRDRP	Controls the dropping of the Data Terminal Ready (DTR) signal associated with locally attached asynchronous lines. The DTRDRP directive does not apply to PRIMOS TCP/IP.
LHC	Assigns a physical device address to an LHC logical controller number. You must include an LHC directive for each LHC in the system. PRIMOS TCP/IP does not require a special controller number or device address. Refer to the section in Chapter 12 entitled Modifying the System Configuration File.
LOGLOG	Allows or disallows use of the LOGIN command while a user is already logged in. No special considerations apply to PRIMOS TCP/IP.
LOTLIM	Specifies the number of minutes allowed for a user to log in. No special considerations apply to PRIMOS TCP/IP.

LOUTQM	Specifies the maximum period of inactivity before PRIMOS automatically logs out users. No special considerations apply to PRIMOS TCP/IP.
NPUSR	Specifies the number of phantoms. Use the formula described in Chapter 12, <i>How to Configure PRIMOS TCP/IP</i> , to calculate the number of phantoms that PRIMOS TCP/IP uses. Add this number to the number of phantoms that all other facilities require.
NTS TIMER	Sets the asynchronous forwarding timer for all LHCs configured to support NTS. No special considerations apply to PRIMOS TCP/IP.
NTSABF	Sets the line I/O buffer sizes for NTS-attached assigned lines. This directive does not apply to PRIMOS TCP/IP because PRIMOS TCP/IP does not support assigned lines.
NTSASL	Specifies the number of buffers to be reserved for NTS-attached assigned lines. PRIMOS TCP/IP does not support assigned async lines. If you have only PRIMOS TCP/IP, set the value for the NTSASL directive to zero. If you have assigned lines on LTSs, refer to Chapter 2 of the <i>NTS Planning and Configuration Guide</i> for guidelines.
NTSBUF	Sets the line I/O buffer sizes for NTS-attached terminal lines. Use the guidelines described in Chapter 2 of the <i>NTS Planning and Configuration Guide</i> to set the value for the NTSBUF directive. No special considerations apply to PRIMOS TCP/IP.
NTSUSR	<p>Specifies the maximum number of NTS users at one time. If the 50 Series system is being accessed only by PRIMOS TCP/IP users but not by LTSs connecting NTS terminal users, set NTSUSR to the maximum number of TELNET users expected at one time. If both PRIMOS TCP/IP and NTS users on LTSs will access the 50 Series system, set NTSUSR to the combined total of the maximum number of NTS and TELNET users expected to access the system at one time.</p> <p>Set this directive to an estimate of the actual number of users rather than to the estimated number of potential users. The number of potential users always exceeds the number of actual simultaneous users; at any given time, only a few users are likely to invoke TELNET on a particular PRIMOS TCP/IP host.</p> <p>Do not set the value for the NTSUSR directive too high. PRIMOS supports a total of 255 processes. This directive reserves some of these processes for NTS users. The combined total for six NTS-related directives (NTSUSR, NTUSR, NPUSR, NSLUSR, NRUSR, and NAMLC) must be less than or equal to 255.</p>

Socket Library Error Messages

This appendix lists the socket library error messages in numeric order. The number of the error appears first, followed by the name and the error message text. The error number and name are defined in the file `SYSCOM>ERROR.H.INS.CC`. The variable *errno* (defined in `stdio.h`) also contains the error number. The error message texts are in the file

`SYSOVL>TUP$_ERROR_TABLE`

<i>Error</i>	<i>Name</i>	<i>Message</i>
17	EACCESS	Permission denied
18	EBADF	Bad socket number
19	EFAULT	Bad address
20	ENOBUFS	No buffer space available
21	EINVAL	Invalid argument
22	ESYSERR	System error occurred
23	EINTR	Interrupted system call
24	ENOPROTOPT	Protocol not available
25	EWouldBLOCK	Operation would block
26	EAFNOSUPPORT	Address family not supported by protocol family
32	ENOTCONN	Socket is not connected
33	EHOSTDOWN	Host is down
34	EHOSTUNREACH	No route to host
35	ENETDOWN	Network is down
36	ENETUNREACH	Network is unreachable
37	EFAILUREADDINGSYNCS	Failure adding socket synchronizers
38	ECONTROLLERDOWN	Interface currently unavailable

<i>Error</i>	<i>Name</i>	<i>Message</i>
39	ESHUTDOWN	Attempted send after shutdown
40	ETIMEDOUT	Connection timed out
41	ECONNREFUSED	Connection refused
42	ECONNRESET	Connection reset
43	ENETRESET	Net reset
44	EUBIERR	Interface problem with controller
45	EMSGSIZE	Invalid message size
46	EISCONN	Socket already connected
47	EOPNOTSUPP	Operation not supported

PRIMOS TCP/IP Glossary

This appendix contains a glossary of terms that refer to TCP/IP, Prime networks, or data communications in general. Terms that are in italics in definitions are defined elsewhere in the glossary.

Access Control List (ACL)

A list of access pairs used to specify the *access rights* of users. An access pair is a pair in the format

identifier:access

where *identifier* is an individual user ID, an access group (beginning with a period), or the identifier \$REST, and *access* is one or more of the following rights:

O	Sets all access rights (except P and ALL)
P	Protects a directory
D	Deletes entries from a directory
A	Adds entries to a directory
L	Reads contents of a directory
U	Attaches to a directory
R	Reads contents of a file
W	Changes contents of a file
X	Executes an Executable Program Format (EPF)
ALL	Allows all rights described above (OPDALURWX)
NONE	Allows no access rights whatsoever

An Access Control List (ACL) protects a file system object by allowing access only to the users listed in the ACL and by allowing those users only the access rights specified in the ACL. You associate an ACL with a file system object using the PRIMOS SET_ACCESS (SAC) command or the PRIMOS EDIT_ACCESS (EDAC) command. See the *PRIMOS User's Guide* for more information on access control lists.

access right

Generally, the degree of access to information that a user or process has on a system or *network*. For example, the Read (R) access right allows a user or process to view a file.

ACL

See Access Control List.

ARPANET

A *Packet Switching Data Network* developed by the Defense Advanced Research Projects Agency (DARPA).

association

A set of five items (*protocol*, local port, local address, remote port, and remote address) that uniquely identifies a pair of communicating processes.

bit

An acronym for binary digit. Eight bits constitute a *byte*.

byte

Eight *bits* of data. A character, for example, is one byte.

Carrier Sense Multiple Access With Collision Detection (CSMA/CD) access method

The access method used to decide which *station* on a *LAN300* takes control of the communication medium and transmits the next message.

CCITT

See Consultative Committee for International Telephony and Telegraphy.

client-server model

The most common form of *socket* communications. The server process listens on a socket at one end of the data path. The server receives connection requests from client processes, with which it exchanges messages.

communication link

The connecting medium between two systems that allows them to transmit and/or receive data. At Rev. 22.0, types of communication links include *LAN300*, *RINGNET*, *synchronous lines* (*full-duplex* and *half-duplex*), and *Packet Switched Data Network* (PSDNs). Two systems can also be linked by one or more intervening *gateway nodes*.

condition

An asynchronous *event*, also called a signal or an exception. User programs set up *on-units* to handle conditions such as a controller failure or a process interrupt.

CONFIG_NTS

A menu-driven and prompt-driven program for configuring *Network Terminal Service* (NTS) networks.

Consultative Committee for International Telephony and Telegraphy (CCITT)

An advisory committee established under the auspices of the United Nations to recommend worldwide communication standards.

control connection

A communication path between a *port* of user FTP and a remote FTP server over which PRIMOS FTP commands are sent.

CSMA/CD

See Carrier Sense Multiple Access With Collision Detection access method.

data connection

A communication path between PRIMOS FTP and a remote FTP server over which data are sent.

data field

The portion of the *data packet* that contains the actual data in *protocol* format.

data packet

See packet, data.

DDN

Defense Data Network, the Department of Defense wide area *Packet Switched Data Network* that includes *ARPANET* and *MILNET* as *subnetworks*.

Distributed Systems Management (DSM)

A set of software products and services that support the administration and day-to-day management of single and networked Prime computer systems. DSM enables systems to be administered and controlled collectively from any convenient point on the *network* and simplifies administrative tasks such as resource monitoring and *event* logging.

DoD

Department of Defense.

domain

The address format (also called the family) of a *socket*. Two of the more important domains are the UNIX domain and the Defense Advanced Research Projects Agency (DARPA) Internet domain.

downline load

The process of reading software and load parameters into a controller that cannot self-boot.

downline load host

The system on a *Network Terminal Service* (NTS) network responsible for loading software into *LAN Terminal Servers 300* (LTS300s) and *LAN Host Controllers 300* (LHC300s) on the network. Normally, both a primary and secondary downline load host are configured to reduce network traffic if the primary host goes down.

DSM

See Distributed Systems Management.

event

A significant system or *network* occurrence such as a cold start, machine check, disk error, or network link problem.

event reporting host

The host on a *LAN300* network responsible for receiving event reports from *LAN Host Controllers 300* (LHCs) and *LAN Terminal Servers 300* (LTSs). Event reports include notices of *downline loads* and upline dumps, along with any kind of error reports. Normally, both primary and secondary event reporting hosts are configured to reduce needless network traffic in case the primary host goes down.

exception

See condition.

File Transfer Protocol (FTP)

Used to copy files across a *network*.

framing

The process of prefixing and suffixing a message with control characters before sending it over a *communication link*. The control characters are said to frame the message.

FTP

See File Transfer Protocol (FTP).

gateway

See gateway node.

gateway node

A system that can route messages between two other systems. Two systems can communicate via a gateway node even if they are not directly connected.

header

The beginning portion of a *packet* that contains information such as the IDs of the target and sending *node* or packet type.

Host Table Utility (TCPHTU)

PRIMOS TCP/IP program that prepares information in TCP/IP configuration files.

International Organization for Standardization (ISO)

Organization responsible for developing Open Systems Interconnection (OSI) model, a 7-layered network architecture.

Internet Protocol (IP)

Standard protocol that enables Prime and other vendors' hosts to communicate over a LAN300. The Internet Protocol (IP) puts data into *packets*, routes the packets to their Internet addresses, and reassembles the packets into data at their destination.

IP

See Internet Protocol.

ISO

See International Organization for Standardization.

LAN

See Local Area Network.

LAN Host Controller 300 (LHC300)

An intelligent controller that is plugged into the backplane of a 50 Series host and provides the interface between PRIMOS and a LAN300. The code on the LHC300 is a combination of Prime written software and the microprocessor. This controller transmits and receives data on a LAN300.

LAN Terminal Server 300 (LTS300)

A remote microprocessor-based asynchronous communications server that provides the LAN300 interface for a maximum of eight asynchronous devices.

LAN300

The Prime IEEE 802.3 compliant *Local Area Network* that uses a bus topology. A standard LAN300 is composed of bus segments, each of which can be a maximum of 500 meters in length. Devices are connected to a LAN300 at a *station*, where a controller, *transceiver*, and tap are attached to the bus segment. A station supports a host or a cluster of terminal or serial printers or both. The LAN300 uses the *Carrier Sense Multiple Access With Collision Detection (CSMA/CD)* access method.

LAN300 Network Management Facility

LAN300 network management software. This facility supports services such as *LAN Host Controller 300 (LHC300)* and *LAN Terminal Server 300 (LTS300)* *downline loading*, LHC300 and LTS300 upline dumping, LHC300 and LTS300 name and address translation, and network event reporting and monitoring. The LAN300 Network Management Facility also provides a user interface that consists of loopback and status commands.

LHC300

See LAN Host Controller 300.

Local Area Network (LAN)

A *network* in which independent computer systems are physically connected and communicate at a high speed over a short distance, such as within a building or building complex. *LAN300* is the Prime IEEE 802.3 compliant LAN that uses a bus configuration.

local system

A system on a *network* from which a user issues commands to communicate with another *remote system*.

LTS300

See LAN Terminal Server 300.

MILNET

A *Packet Switched Data Network* developed by the Defense Advanced Research Projects Agency (DARPA).

MIL STD

Military Standard. The official military version of a specification.

network

A group of independent computer systems that are connected by communication media such as PSDNs, LANs, and *synchronous lines* and that communicate and share resources. A network can consist of systems that are all physically connected and communicate over short distances, as in a LAN, or of systems that use different communication media to communicate over long distances.

Network Administrator

The person responsible for maintaining the proper and continuous operation of a *network*. Tasks include configuring the network, ensuring that appropriate security measures are taken, and maintaining the daily network operation. Sometimes, the same person serves as Network Administrator and *System Administrator*. *See also* System Administrator.

network configuration

A description of the systems, services, and communication media that make up a *network*.

network configuration file

The file that contains the network configuration in binary format. The TCP/IP configuration file is named HOSTS.TXT. The *Network Administrator* creates this file with a standard Prime text editor such as ED or EMACS.

network protocol

See protocol.

Network Terminal Service (NTS)

A collection of software designed to support communications between terminals and hosts over a *LAN300 network*.

NIC

DDN Network Information Center, which is located at SRI International, Menlo Park, California.

node

An independent computer system that is part of a *network*.

NTS

See Network Terminal Service.

NTS configuration file

A segment directory that contains the NTS configuration in binary format. This file is created with CONFIG_NTS, the NTS configuration program, and loaded into an NTS host with the START_NTS command.

on-unit

A user-defined subroutine that handles asynchronous *events (conditions)*.

out-of-band data

High priority messages sent from one process to another.

packet

A sequence of data and control characters that are arranged in a specific format and transmitted as a unit.

packet, data

A package that contains *packet protocol* and data. It consists of a leading frame, a header, a data field, and a trailing frame.

packet size

The number of *bytes* in a *packet*.

Packet Switching Data Network (PSDN)

A type of public data network that offers wide area communications to its subscribers.

path

The sequence of intervening systems between two given systems in a *network*.

peer

The *socket* at the opposite end of a connection.

port

An address within a *node* to which an incoming *network* request can be routed.

PRIMENET

The Prime distributed networking software that offers local and wide area networking facilities. PRIMENET runs on a number of physical transmission media: *RINGNET*, *LAN300*, *PSDNs*, and synchronous lines.

PRIMENET address

A numeric address that *PRIMENET* uses internally to identify a *node*. *CONFIG_NET* generates this address based on the name of the node.

privileged connection

A *socket* connection that has a port number less than 1024.

protocol

A set of rules governing communication between two systems in a *network*.

PSDN

See Packet Switching Data Network.

remote system

A system that can communicate with the *local system* through a *network*. A remote system can be attached to the same LAN as a local system.

remote user

A user on a *remote system*.

RFC

Request for Comments. Technical notes that describe *DDN* protocols and networking specifications.

RINGNET

Prime local area network that uses a ring topology.

server

A cooperating set of processes available to perform one or more functions.

signal

See condition.

SMTP

Simple Mail Transfer Protocol.

socket

An endpoint for communication between processes. The PRIMOS socket implementation enables a user program on a 50 Series system to communicate with another application running on a remote host over an IEEE 802.3 *network* using the *Transmission Control Protocol* (TCP) and the *User Datagram Protocol* (UDP).

SRI

SRI International, Menlo Park, California, location of the *DDN* Network Information Center (NIC).

START_DSM

Command that brings up *Distributed Systems Management* (DSM) on the system.

START_NTS

Command that starts up the *Network Terminal Service* (NTS).

START_TCP

Command that starts up TCP/IP *protocols* on an LHC300.

station

A point on a *LAN300* at which a controller, *transceiver*, and tap are attached to a bus segment.

STOP_DSM

Command that shuts down *Distributed System Management* (DSM) by logging out all DSM server processes.

STOP_TCP

Command that stops all TCP/IP-related processes.

subnet mask

A bit field in a TCP/IP file (THISHOST) that identifies a host as belonging to a *subnetwork*.

subnetwork

A subset of systems on a *network* directly connected to a set of systems in a larger network. PRIMOS TCP/IP supports subnetworks.

System Administrator

The person responsible for maintaining the proper and continuous operation of a system. The System Administrator's duties can include network-related tasks such as setting PRIMOS FTP-related access rights, and setting up PRIMOS TCP/IP. At times, the same person serves as System Administrator and *Network Administrator*. *See also* Network Administrator.

TCP

See Transmission Control Protocol (TCP).

TCPHTU

See Host Table Utility.

TELNET protocol

Standard network *protocol* that enables a local user to log in to a *remote system* and a *remote user* to log in to a 50 Series system.

transceiver

A device that receives and transmits data simultaneously.

Transmission Control Protocol (TCP)

Standard network *protocol* that corresponds to the transport layer of *ISO/OSI* Model.

UDP

See User Datagram Protocol.

upline dump host

A Prime host on a *Network Terminal Service (NTS) network* responsible for receiving upline dumps from *LAN Host Controllers 300 (LHC300s)* and *LAN Terminal Servers 300 (LTS300s)* on the network. Upline dumps are used to diagnose software failures. Normally, both a primary and secondary upline dump host are configured to reduce network traffic if the primary host goes down.

User Datagram Protocol (UDP)

A transaction-oriented *protocol* that supports the transmission of unsequenced segments of data to an Internet destination system.

wildcard gateway

A *gateway* system that can route data to either a final destination or another gateway when the address of the gateways or subnetworks on the destination route are not known.

Symbols

- ! command (PRIMOS TCP/IP), 4-3
- ? command (PRIMOS TCP/IP), 4-5

A

- A command (PRIMOS TCP/IP), 3-23, 4-2
- Abbreviating commands, 3-5, 4-2
- ABORT command (PRIMOS TCP/IP), 5-5
- Access rights, 1-5
 - for installing PRIMOS TCP/IP, 11-4, 11-14
 - for remote system users, 5-2
 - for retrieving files, 3-3, 5-2
 - for sending files, 3-2, 5-2
 - for specific FTP commands, 3-29
 - for the MGET command, B-2
 - for the MPUT command, B-2
 - list of, 3-2
 - modified by NTS, 11-16
 - on the SAD, 11-15
 - on the TCP/IP* directory, 11-16
 - required for deleting files, 5-2
 - resetting on the SAD, 5-3, 11-15, 14-10
 - setting, 3-4
 - tables, 3-3
 - to a remote 50 Series system, 3-3
 - to parent directories, 3-3
- ACCOUNT command (PRIMOS TCP/IP), 3-27, 4-6, 5-5
- Account numbers, 3-10, 4-10, 4-18
- ACL groups
 - .NETWORK_MGT\$, E-2
 - .TCP_FTP\$, 1-5, 5-1
 - .UBI\$, 8-13, 9-10, 9-13, B-2, B-5
 - .WS1_FTP\$, 5-1
- Add-gateway parameter, 13-12
- Add-route parameter, 13-12
- Add-transport parameter, 13-12
- Add-x25-address parameter, 13-11
- Alias, 12-7
- AMLCLK directive, F-1
- AMLTIM directive, F-1
- ARPANET, 1-8, 12-5

ASCII

- file type, 3-12, 4-17
- files, 1-4, 5-8
- mailing files, 7-6
- Prime, 3-12, 4-6
- standard, 3-12, 4-6
- ASCII command (PRIMOS TCP/IP), 3-27, 4-6
- ASSIGN command (PRIMOS), 11-4
- Assigned lines, F-1 to F-2
- ASYNCH JUMPER directive, F-1
- Asynchronous lines, F-1
- ATTACH command (PRIMOS), 4-2 to 4-4, 5-5

B

- Berkeley Software Distribution (BSD), 8-1
- Binary
 - data, 3-12
 - file type, 3-12, 4-17
 - files, 1-4, 4-6, 4-9 to 4-10, 4-16, 5-8
 - mailing files, 7-6
- BINARY command (PRIMOS TCP/IP), 3-13, 3-27, 4-6
- Blocking sockets, 9-16
- Block-primenet parameter, 13-11
- Broadcast address, 9-55
- BYE command (PRIMOS TCP/IP), 3-27, 4-7, 4-15

C

- Carrier Sense Multiple Access With Collision Detection access method, 1-3
- CD command (PRIMOS TCP/IP), 3-25, 3-27, 4-7
- CDUP command (PRIMOS TCP/IP), 5-5
- CHDIR command (PRIMOS TCP/IP), 3-23, 4-3
- Client-server model, 8-2
- CLOSE command (PRIMOS TCP/IP), 2-6, 3-15, 3-19, 3-26 to 3-27, 4-7

- CMDNCO directory, 11-3, 11-7 to 11-8, 12-17
- FTP.RUN file, 11-8
- MAIL.RUN file, 11-8
- START_FTP_SERVER.RUN file, 11-8
- START_MAILER.RUN file, 11-8
- START_SMTP.RUN file, 11-8
- START_TCP/IP.RUN file, 11-8
- STOP_MAILER.RUN file, 11-8
- STOP_TCP/IP.RUN file, 11-8
- TELNET.RUN file, 11-8
- Command mode, 2-1
- Command-line editor, 3-7
- COMM_CONTROLLER command (PRIMOS), 14-5, 14-16 to 14-17, 15-4, 15-14
 - error messages, D-2
 - examples of, 14-7, 15-4, 15-7, 15-11, 15-14
 - format, 14-5
 - options, 14-6
 - subcommands, 14-5
- Condition handlers, 8-22, 9-7
- CONFIG file
 - see*: Configuration files
- CONFIG_NTS command (NTS), 12-19, 12-24, 15-3, 15-10, 15-14
- CONFIG_UM utility (DSM), 16-3
- Configuration directives, 11-3
 - NTS-related, F-1
- CONFIGURATION file, 13-7
- Configuration files
 - examples of, 12-6, 12-10, 15-1, 15-5, 15-9, 15-13
 - HOSTNAME_CONFIG, 12-2
 - HOSTS.TXT, 12-5, 12-24
 - Network Terminal Service, 12-19, 15-3, 15-6, 15-10, 15-14
 - PRIMOS TCP/IP, 3-7, 12-11
 - system, 12-17, 12-24, 15-6
 - THISHOST, 12-1, 12-11
- CONFIGURATION.TEMPLATE file, 11-12, 13-4
- CONTROL-P command (PRIMOS TCP/IP), 3-27, 4-7

CREATE_MAILBOX command
 (PRIMOS TCP/IP), 13-19
 CSMA/CD, 1-3
 CWD command (PRIMOS TCP/IP),
 3-25, 3-27, 4-8, 5-5

D

Data connection, 5-6, 14-2, B-2, B-9
 to B-10
 Data structures
 file, 4-17, 5-8
 record, 4-17, 5-8
 Data Terminal Ready (DTR), F-1
 DDN Protocol Handbook, 12-5
 DEAD.LETTER file, 6-7, 6-12, 13-3
 DEBUG command (PRIMOS
 TCP/IP), 2-7
 DEFAULTLOG file, 16-3
 DELE command (PRIMOS TCP/IP),
 5-5
 DELETE command (PRIMOS
 TCP/IP), 3-27, 4-8
 Device addresses
 displayed by
 LIST_LHC_STATUS
 command, 16-6
 in THISHOST file, 12-11, 15-3,
 15-14
 read by Host Table Utility, 12-17
 DIR command (PRIMOS TCP/IP),
 3-25, 3-27, 4-9
 Directives
 AMLCLK, F-1
 AMLTIM, F-1
 ASYNC, F-1
 DISLOG, F-1
 DTRDRP, F-1
 LHC, F-1
 LOGLOG, F-1
 LOTLIM, F-1
 LOUTQM, F-2
 NPUSR, F-2
 NTS TIMER, F-2
 NTSASL, F-2
 NTSBF, F-2
 NTSBUF, F-2
 NTSUSR, F-2
 SYSNAM, 12-23

Directories
 CMDNC0, 11-3
 DOWN_LINE_LOAD*, 11-3,
 11-7 to 11-8, 14-7
 HELP*, 11-3, 11-7, 11-11
 INFO, 11-3, 11-7, 11-12
 LIB, 11-3, 11-7, 11-12
 LIBRARIES*, 11-7, 11-12
 LIBRARIES*>MAPS, 11-7, 11-12
 MAILER*, 11-3, 11-7
 MAILER*>DB, 11-7, 11-12
 MAILER*>EXEC, 11-7, 11-13
 MAILER*>LOG, 11-7, 11-13
 MAILER*>MBOX, 11-7, 11-13
 MAILER*>QUEUE, 11-7, 11-13
 SYSCOM, 11-3, 11-7, 11-13
 SYSOVL, 11-3, 11-7, 11-14
 TCP/IP, 11-3
 TCP/IP*, D-2, 11-3, 11-7, 11-9,
 12-11
 TCP/IP*>COMI, 11-9
 TCP/IP*>ISC, 11-9
 TCP/IP*>LOG_FILES, 11-7, 11-9
 TCP/IP*>LOG_
 FILES>ARCHIVE, 11-9
 TCP/IP*>MESSAGE_TEXT,
 11-7, 11-10
 TCP/IP*>RUN, 11-7, 11-10
 TCP/IP*>SMTP, 11-7, 11-10
 TCP/IP*>SMTP>INCOMING,
 11-10
 TCP/IP*>SMTP>OUTGOING,
 11-11
 Disctime, F-1
 DISLOG directive, F-1
 Distributed Systems Management
 (DSM), 14-5, 14-17
 CONFIG_UM utility, 16-3
 configuring, 12-1
 DEFAULT.LOG file, 16-3
 DSM_LOGFILE file, 16-3
 log files, 14-16, 16-3
 unsolicited messages, 12-1
 DLL.LOG file, 16-3
 Domain names, 6-2, 12-7
 Domain parameter, 13-4
 Domains, 8-2
 AF_INET, 8-2
 AF_UNIX, 8-2
 Dot notation, 12-3

Downline loading, 12-22, 14-5
 DOWN_LINE_LOAD* directory,
 11-3, 11-7 to 11-8
 LHC300_TCPDL file, 11-8, 14-7
 DSM
 see: Distributed Systems
 Management
 DSM_LOGFILE file, 14-16, 16-3
 DTRDRP directive, F-1

E

EBCDIC file type, 4-18
 EDIT_CMD_LINE command
 (PRIMOS), 2-4, 3-7, 3-11
 EDIT_PROFILE utility, 5-3, 11-15,
 14-10
 ENTRY\$ search rule, 11-17
 ENTRY\$.SR file, 11-17
 Error messages
 COMM_CONTROLLER
 command, D-2
 configuration and installation, D-1
 FTP server process, B-9
 LIST_LHC_STATUS command,
 E-1
 MAIL, C-1
 network management, E-1
 socket, G-1
 socket, displaying, 8-28, 9-37
 User FTP, B-1
 User TELNET, A-1
 Escape character, 2-3
 ESCAPE command (PRIMOS
 TCP/IP), 2-5
 Event reporting hosts, 12-22
 ! command (PRIMOS TCP/IP), 4-3

F

File structure, 4-17, B-12
 File system objects, 7-5
 File Transfer Protocol (FTP), 1-1 to
 1-2, 1-4, 1-8
 File types, 4-15 to 4-17
 ASCII, 3-12, 4-17, 5-8
 binary, 3-12
 EBCDIC, 4-17
 Image, 4-17, 5-8
 local, 3-12
 selecting, 3-12

Files

- ASCII, 1-4, 5-8
- binary, 1-4, 5-8
- changing names of, 4-16
- overwriting, 4-15 to 4-17
- renaming, 5-7
- retrieving, 3-18
- sending, 3-14

FTP

- connecting to a remote server, 3-9
- displaying help, 3-25
- exiting, 3-26
- invoking, 3-7
- obtaining directory information, 3-22
- retrieving files, 3-18
- selecting file types, 3-12
- sending files, 3-15
- troubleshooting, 3-11

FTP buffer, 3-14, 4-6

FTP command (PRIMOS TCP/IP), 3-7

FTP commands

- !, 4-3
- ?, 3-23, 4-5
- A, 3-23, 4-2
- abbreviating, 3-5, 4-2
- ACCOUNT, 3-27, 4-6
- ASCII, 3-27, 4-6
- BINARY, 3-13, 3-27, 4-6
- BYE, 3-27, 4-7
- CD, 3-25, 3-27, 4-7
- CHDIR, 3-23, 4-3
- CLOSE, 3-15, 3-19, 3-26 to 3-27, 4-7
- CONTROL-P, 3-27, 4-7
- CWD, 3-25, 3-27, 4-8
- DELETE, 3-27, 4-8
- DIR, 3-25, 3-27, 4-9
- GET, 3-18, 3-27, 4-9
- HASH, 3-23, 4-4
- HELP, 3-25, 4-4
- IMAGE, 3-27, 4-10
- LCD, 3-23, 4-4
- LD, 3-23, 4-5
- LIST, 3-25, 3-27, 4-10
- LOG, 3-9, 3-27, 4-10, 4-18
- LS, 3-25, 3-27, 4-10
- MGET, 3-20, 3-27, 4-11
- MKDIR, 3-27, 4-12
- MODE, 3-27, 4-13

MPUT, 3-27, 4-13

NLST, 3-25, 3-28, 4-14

OPEN, 3-9, 3-28, 4-14

PUT, 3-28, 4-15

PWD, 3-25, 3-28, 4-15

QUIT, 3-26, 3-28, 4-15

RECV, 3-28, 4-15

RENAME, 3-28, 4-16

RETRIEVE, 3-28, 4-16

RMDIR, 3-28, 4-16

SEND, 3-14, 3-28, 4-17

SHOW, 3-23, 4-5

STATUS, 3-13, 3-23, 4-5

STORE, 3-28, 4-17

STRUCTURE, 3-28, 4-17

summary of (tables), 3-26

TYPE, 3-13, 3-28, 4-17

USER, 3-9, 3-12, 3-28, 4-18

FTP HELP facility, 1-5, 3-25

FTP messages, B-1

FTP server commands

ABORT, 5-5

ACCOUNT, 5-5

CDUP, 5-5

CWD, 5-5

DELE, 5-5

LIST, 5-5, B-11

MKD, 5-5

MODE, 5-6

NLST, 5-6

NOOP, 5-6

PASS, 5-6, B-10, B-13

PORT, 5-6

PWD, 5-6

QUIT, 5-6

RETR, 5-7

RMD, 5-6

RNFR, 5-7, B-10 to B-11

RNTQ, 5-7, B-10 to B-11

STOR, 5-7

STRU, 5-7

TYPE, 5-8

USER, 5-8, B-12 to B-13

FTP server process

name of spawned phantom, 14-11

phantoms spawned, 14-2

stopping, 14-15

FTP server (remote), 3-9, 5-6, 5-8

program messages, B-9

FTP session, exiting, 4-15

FTP.HELP file, 11-11

FTPRUN file, 11-8

FTP_SERVER nn process, 12-18

FTP_SP_LOG file, 16-2

FTP_SS_LOG file, 16-2

FTP_UP_LOG file, 16-2

G

Gateway addresses, 14-9

Gateways, 1-8, 12-5, 12-7, 12-9, 12-12

configured as hosts, 12-7

in the HOSTS.TXT file, 12-7

Gateway-scan-interval parameter, 13-8

GATEWAYS_ nn file, 12-17

GET command (PRIMOS TCP/IP), 3-18, 3-27 to 3-28, 4-9, 4-16

Gracetime, F-1

H

HASH command (PRIMOS TCP/IP), 3-14, 3-23, 4-4

Hashed directories, mailboxes, 13-19

HELP command (PRIMOS TCP/IP), 2-7, 3-25, 4-4 to 4-5

HELP* directory, 11-3, 11-7, 11-11

FTP.HELP file, 11-11

MAIL.HELP file, 11-11

START_FTP_SERVER.HELP file, 11-11

START_MAILER.HELP file, 11-11

START_SMT.HELP file, 11-11

START_TCP/IP.HELP file, 11-11

STOP_MAILER.HELP file, 11-11

STOP_TCP/IP.HELP file, 11-11

TCP/IPHTU.HELP file, 11-11

TELNET.HELP file, 11-11

Help facility, 1-5, 3-25

Host cache, 12-15

Host names

DoD, 12-7

domain style, 12-7

PRIMOS, 12-7

Host Table Utility (TCP/IPHTU), 12-1, 12-16, 12-24

error conditions, 12-17

running, 12-16

HOST.CACHE file, 13-3, 13-18
Hostname parameter, 13-13
Hostname Service, 3-7
Hostname Service protocol, 1-6,
 9-54, 12-14, 13-3
HOSTNAME_CACHE file, 12-16
HOSTNAME_CONFIG file, 12-2,
 15-14
 description of, 12-14
 examples of, 15-3, 15-6, 15-10
 format of, 12-15
 how to update, 14-14
HOSTNAME_LOG file, 14-9, 16-2
HOSTS file, 12-17
HOSTS.TEMPLATE file, 11-9
HOSTS.TXT configuration file,
 12-11, 12-17, 12-24
 description of, 12-1, 12-5
 examples of, 15-5, 15-9, 15-13
 how to update, 14-14
 meaning of line entries, 12-6

I

ICS controllers, D-2
IEEE 802.3, 1-3
IMAGE command (PRIMOS
 TCP/IP), 3-27, 4-10
Image file type, 4-17
INADDR_ANY, 8-21 to 8-22, 9-20
Include files, 8-27 to 8-28, 9-4
 see also: SYSCOM directory
INET.H.INS.CC file, 8-27, 11-13
INFO directory, 11-3, 11-7, 11-12
 TCP/IP.RUNO file, 11-12
IN.H.INS.CC file, 8-27, 11-13
Installation procedures
 for LAN300 Network
 Management software, 11-3
 for Network Terminal Service
 (NTS), 11-3
 for PRIMOS TCP/IP, 11-3
International Organization for
 Standardization, 1-2
Internet addresses, 1-6
 assigning, 12-11
 description of, 12-3
 examples of, 12-6, 12-12, 12-14,
 15-3, 15-14
 gateways, 12-7

 in socket names, 8-4
 in the HOSTS.TXT file, 12-5
 passed to the controller, 14-9
 processed by Host Table Utility,
 12-17
Internet Host Table, 12-5
Internet Protocol (IP), 1-1 to 1-2
InterServer Communications, 1-6,
 8-25 to 8-26
IOCTL.H.INS.CC file, 8-27, 11-14
IPPROTO_IP socket level, 8-30
IPPROTO_TCP socket level, 8-30
ISC
 see: InterServer Communications
ISO (International Organization for
 Standardization), 1-2
ISO/OSI Reference Model, 1-2

L

LAN Host Controller 300 (LHC),
 1-1, 1-4, 1-7, 11-8, 12-11
LAN Terminal Server 300 (LTS),
 12-21, 15-3, 15-14
LAN300 and PRIMENET, 1-7
LAN300, description of, 1-1 to 1-2,
 1-4
LAN300 Network Management
 configuring, 12-22
 error messages, E-1
 installation procedures, 11-3
 LAN Terminal Server 300, 12-22
 server, 12-19, 14-9
LCD command (PRIMOS TCP/IP),
 3-23, 4-4
LD command (PRIMOS TCP/IP),
 3-23, 4-5
LHC directive, 1-6, F-1
 added to CONFIG file, 12-18
 examples of, 15-2, 15-6, 15-9,
 15-13
LHC number, 16-5, E-4
LHC300_TCP.DL file, 11-8, 14-7
LHC_DLL_SERVER process, 14-8
LHC_hostname-deviceaddr.date.time,
 16-1
LHC
 see: LAN Host Controller 300
LIB directory, 11-3, 11-7, 11-12
 SOCKET.BIN file, 11-12

LIBRARIES* directory, 11-3, 11-7,
 11-12
 SOCKET_IX.RUN file, 11-12,
 11-17
 SOCKET.RUN file, 11-12, 11-17
LIBRARIES*>MAPS directory, 11-7,
 11-12
 SOCKET_IX.MAP file, 11-12
 SOCKET.MAP file, 11-12
LIST command (PRIMOS TCP/IP),
 3-25, 3-27, 4-10, B-11
LIST_ACCESS command
 (PRIMOS), 3-4, 11-4
LIST_LHC_STATUS command
 (PRIMOS), 16-4
 error messages, E-1
 format, 16-4
Load balancing, 1-8
Local commands, 4-2
Local-area-interval parameter, 13-8
Local-area-retry parameter, 13-8
Local-area-search parameter, 13-9
LOG command (PRIMOS TCP/IP),
 3-9, 3-27, 4-10, 4-18
Log files
 FTP_SP_LOG, 16-2
 FTP_SS_LOG, 16-2
 FTP_UP_LOG, 16-2
 HOSTNAME_LOG, 14-9, 16-2
 MAILER*>LOG directory, 11-13
 MAIL.LOG, 16-3
 MANAGER_LOG, 14-9, 16-2
 PORTAL_LOG, 14-10, 16-2
 SERVER_FTP_LOG, 14-10, 16-3
 SMTP_LOG, 14-11, 16-3
 START_TCP/IP_LOG, 14-9, 16-3
 STOP_TCP/IP_LOG, 14-15
 TCP/IP*>LOG_FILES directory,
 11-9
Log-file-reset parameter, 13-9
Logical controller numbers, 12-18,
 16-4
LOGIN command (PRIMOS), F-1
LOGLOG directive, F-1
LOTLM directive, F-1
LOUTQM directive, F-2
LS command (PRIMOS TCP/IP),
 3-25, 3-27, 4-10
LTS
 see: LAN Terminal Server 300

M

MAGRST command (PRIMOS),
11-3

MAIL

- appending messages, 7-8 to 7-9
- certifying, 7-4
- configuring, 13-1
- copying messages, 7-8
- creating mailboxes, 6-3, 13-19
- deleting messages, 7-9
- displaying help text, 7-11
- error messages, C-1
- help facility, 6-11
- invoking, 6-4
- listing messages, 6-4, 7-11
- managing mailboxes, 7-1
- names of servers, 6-3
- node and routing parameters,
13-11
- optional parameters, 13-7
- printing messages, 7-11
- reading messages, 6-5
- replying to messages, 7-12
- required configuration parameters,
13-4
- restoring deleted messages, 7-14
- routing messages, 13-3
- running over PRIMENET, 13-2
- saving messages, 6-8
- sending a file from command
level, 6-7
- sending binary files, 7-6
- sending copies of mail, 7-4
- sending directories, 7-5
- sending messages, 6-6, 7-13
- sending non-ASCII files, 7-5
- sending to alias names, 7-3
- specifying a subject, 7-7
- starting servers, 14-11
- terminating mail session, 7-12
- using a mailing list, 7-6

MAIL command (PRIMOS TCP/IP),
7-1

MAIL configuration file, how to
update, 13-18

MAIL configuration parameters

- add-gateway, 13-12
- add-route, 13-12
- add-transport, 13-12
- add-x25-address, 13-11

- block-primenet, 13-11
- domain, 13-4
- gateway-scan-interval, 13-8
- hostname, 13-13
- local-area-interval, 13-8
- local-area-retry, 13-8
- local-area-search, 13-9
- log-file-reset, 13-9
- message-timeout, 13-9
- postmaster, 13-7
- relay node, 13-6
- transport-scan-interval, 13-8
- wide-area-interval, 13-9
- wide-area-retry, 13-9
- wide-area-search, 13-10

MAIL ID, 6-2

MAIL interactive commands, 7-7

- APPEND, 6-9, 7-8
- COPY, 7-8
- DELETE, 7-9
- FILE, 6-8, 7-9
- HELP, 7-11
- LIST, 7-11
- PRINT, 7-11
- QUIT, 7-12
- REPLY, 7-12
- SEND, 7-13
- UNDELETE, 7-14

MAIL servers

- MAILER_DAEMON, 6-3, 14-2
- SMTP_SENDER0, 6-3, 14-2
- SMTP_SERVERnn, 6-3, 14-2

MAIL subcommands

- !, 7-6
- ALIAS, 7-3
- BINARY, 7-4
- CANCEL_FORWARDING, 7-1
- CC, 7-4
- CE, 7-4
- DELETE_MAILBOX, 7-2
- FILE, 7-5
- HELP, 7-2
- INTERACTIVE, 7-2
- LIST, 7-2
- SET_FORWARD_TO, 7-2
- SUBJECT, 7-7
- TO, 7-7

Mailboxes, 7-1

- ACL rights on, 13-19
- created as hashed directories,
13-19

- error messages, 6-3
- how to create, 13-18
- initializing, 6-3
- preserved when TCP/IP
reinstalled, 11-13

MAILER* directory, 11-3, 11-7

MAILER.CPL file, 11-13

MAILER_DAEMON, 6-3, 13-3,
14-2, 14-12

MAILER*>DB directory, 11-7,
11-12

CONFIGURATION file, 13-7

CONFIGURATION.TEMPLATE
file, 11-12, 13-4

MAILER*>EXEC directory, 11-7,
11-13

MAILER.CPL file, 11-13

START_MAILER.RUN file,
11-13

STOP_MAILER.RUN file, 11-13

MAILER*>LOG directory, 11-7,
11-13

MAILER*>MBOX directory, 11-7,
11-13

MAILER*>QUEUE directory, 11-7,
11-13

MAIL.HELP file, 11-11

MAIL.LOGnn files, 16-3

MAIL.RUN file, 11-8

MANAGER_LOG log file, 14-9,
16-2

MAXUSR command (PRIMOS),
14-16, 14-18

Message-timeout parameter, 13-9

MGET command (PRIMOS TCP/IP),
3-20, 3-27, 4-11

- access rights, B-2

MKD command (PRIMOS TCP/IP),
5-5

MKDIR command (PRIMOS
TCP/IP), 3-27, 4-12

Mode

see: STREAM mode

MODE command (PRIMOS TCP/IP),
3-14, 3-27, 4-5, 4-13

MPUT command (PRIMOS TCP/IP),
3-27, 4-13

- access rights, B-2

N

Names

see: Host names

Naming conventions

see: Host names

NETDB.H.INS.CC file, 8-27, 11-14

Network Information Center (NIC), 12-5

Network management

see: LAN300 Network

Management

Network Management Extension (NME), E-3

Network management log files

DLL.LOG, 16-3

NMSR.LOG, 16-3

ULD.LOG, 16-3

Network names, 15-1, 15-12

Network numbers, 12-11, 15-1, 15-10, 15-12

Network Terminal Service (NTS), 1-4, 1-7, 12-1

CONFIG_NTS command, 12-19

configuration file, 14-9, 15-3,

15-6, 15-10, 15-14

configuring, 12-19

main menu, 12-20

START_NTS command, 14-9

unconfigured nodes, 12-21, 15-6

.NETWORK_MGT\$ ACL group, E-2

NETWORK_MGT* directory, 11-17, 16-3

NETWORK_MGT.INSTALL.CPL file, 11-3

NIC

see: Network Information Center

Nicknames, use of in configuration file, 12-7

NLST command (PRIMOS TCP/IP), 3-25, 3-28, 4-14

NME

see: Network Management Extension

NM_SERVER process, 14-3, 14-9, 14-12, 16-3, E-4

NMSR.LOG file, 16-3

Node address

long form, E-1

short form, E-1

Node Status database, E-4, E-6

Nodes

local, 1-6

PRIMOS name, 12-12

unconfigured, 12-21, 15-6

Nonblocking sockets, 9-16

NOOP command (PRIMOS TCP/IP), 5-6

NPUSR directive, 12-18, F-2

NTS configuration file, examples of, 15-3, 15-6, 15-10, 15-14

NTS* directory, NTS.CONFIG file, 14-9

NTS protocols, 14-7

NTS TIMER directive, F-2

NTSABF directive, F-2

NTSASL directive, F-2

NTS*>NTS.CONFIG file, 14-17

NTSBUF directive, F-2

NTS.CONFIG file, 14-9

NTS.INSTALL.CPL file, 11-3

NTS

see: Network Terminal Service

NTSUSR directive, F-2

O

On-units, 8-22, 9-7

OPEN command (PRIMOS TCP/IP), 2-6, 3-9, 3-28, 4-14

Operational state, 16-6

OPTIONS command (PRIMOS TCP/IP), 2-7

P

PASS command (PRIMOS TCP/IP), 5-6, 5-8, B-10

PASSWORD command (PRIMOS TCP/IP), B-13

PASSWORD_DIRS command (PRIMOS), 13-18

Password-protected directories, 3-15, 5-4

attaching to, 4-2, 4-4

Passwords, 1-5, 3-15, 5-6

protecting directories, 5-4

required by remote server, 4-10, 4-18

using FTP, 3-4

Pathnames, 4-15 to 4-16, 5-5 to 5-7

full, 5-5 to 5-6

relative, 5-5 to 5-6

Peer, 9-26

Performance screen, 16-4

Phantoms, 12-18

TCPFTP_SERVER_

PHANTOM_{nn}, 14-2, 14-11

Physical device addresses

see: Device addresses

Port addresses, 5-6, B-11

PORT command (PRIMOS TCP/IP), 5-6

PORTAL_LOG file, 14-10, 16-2

Ports, 8-4

reserved, 8-13

Postmaster parameter, 13-7

Prime LAN300 prefix, E-1, E-5

PRIMENET, 6-1, 13-2

node name, 15-9

PRIMOS node name, 12-12

PRIMOS socket library

allocation of sockets, 14-13

calls supported, 8-5

differences from UNIX, 8-21

handling signals, 8-22

include files, 8-27, 9-4

network library subroutines, 8-7

prerequisites, 8-1

restrictions, 8-30

routines summarized, 9-1

sample programs, 8-8, 8-19

sharing sockets, 8-26

socket environment subroutines, 8-5

socket subroutines, 8-6

using an application, 8-8

PRIMOS socket routines

TUP\$accept(), 8-17, 9-16

TUP\$bind(), 8-15, 9-19

TUP\$close_socket(), 8-19, 9-22

TUP\$close_socket_env(), 8-19, 9-5

TUP\$connect(), 8-15, 9-23

TUP\$getalladdrs(), 9-54

TUP\$getbroadcastaddr(), 9-55

TUP\$getdevaddrs(), 8-12, 9-57

TUP\$gethostbyaddr(), 9-60

TUP\$gethostbyname(), 9-63

TUP\$gethostname(), 9-65

PRIMOS socket routines (continued)

TUP\$getpeername(), 9-26
 TUP\$getsockname(), 9-28
 TUP\$getsockopt(), 9-30
 TUP\$inet_addr(), 9-66
 TUP\$inet_lnaof(), 9-67
 TUP\$inet_makeaddr(), 9-68
 TUP\$inet_netof(), 9-69
 TUP\$inet_network(), 9-70
 TUP\$inet_ntoa(), 9-71
 TUP\$ioctl(), 9-33
 TUP\$listen(), 8-16, 9-35
 TUP\$pass_socket(), 8-26, 9-7
 TUP\$perror(), 9-37
 TUP\$receive_passed_socket(),
 8-26, 9-9
 TUP\$recv(), 8-17, 9-38
 TUP\$recvfrom(), 8-17, 8-20, 9-38
 TUP\$reset_socket_env(), 8-22,
 9-10
 TUP\$select(), 9-41
 TUP\$send(), 8-17, 9-43
 TUP\$sendto(), 8-17, 8-20, 9-43
 TUP\$setsockopt(), 9-46
 TUP\$setup_socket_env(), 8-13,
 9-13
 TUP\$shutdown(), 9-49
 TUP\$socket(), 8-14, 9-51
 PRIMOS system names, 12-7 to 12-8
 PRIMOS TCP/IP
 architecture, 1-1
 configuration files, 12-1
 directories and files, 11-7
 FTP, 3-1
 installation procedures, 11-1
 MAIL, 6-1
 operator facilities, 1-6
 restrictions and limitations, 1-8
 socket subroutine library, 8-1, 9-1
 User TELNET, 2-1
 PRIMOS TCP/IP configuration file,
 12-11
 PRIMOS TCP/IP Host Table Utility
 (TCP/IPHTU), error
 messages, D-1
 PRIMOS TCP/IP phantoms,
 TCPFTP_SERVER_
 PHANTOM nn , 14-2, 14-11
 PRIMOS TCP/IP runfiles
 FTP.RUN, 11-8

MAIL.RUN, 11-8
 SMTP.RUN, 11-10
 SOCKET_IX.RUN, 11-12
 SOCKET.RUN, 11-12
 START_FTP_SERVER.RUN,
 11-8
 START_MAILER.RUN, 11-8,
 11-13
 START_SMTP.RUN, 11-8
 START_TCP/IP.RUN, 11-8
 STOP_MAILER.RUN, 11-8,
 11-13
 STOP_TCP/IP.RUN, 11-8
 TCPFTPSP.RUN, 11-10
 TCPFTPSS.RUN, 11-10
 TCP/IP_DUMP.RUN, 11-10
 TCP/IP_MANAGER.RUN, 11-10
 TCP/IPHTU.RUN, 11-9
 TELNET.RUN, 11-8
 PRIMOS TCP/IP user process,
 stopping, 14-15
 PRIMOS.COMI file, modifying,
 14-16
 Privileged connections, 8-13, 9-10,
 9-13

Program messages

see: Error messages

Project IDs, 4-6, 5-5, 14-10
 for users of server FTP, 5-3
 insufficient access to SAD, 5-3
 resetting ACLs on the SAD,
 11-15, 14-10
 PUT command (PRIMOS TCP/IP),
 3-28, 4-15, 4-17
 PWD command (PRIMOS TCP/IP),
 3-25, 3-28, 4-15, 5-6

Q

? command (PRIMOS TCP/IP), 3-23,
 4-5
 QUIT command (PRIMOS TCP/IP),
 2-7, 3-26, 3-28, 4-8, 4-15,
 5-6

R

Record structure, 5-8
 RECV command (PRIMOS TCP/IP),
 3-28, 4-15

REDEFINE.H.INS.CC file, 8-5, 8-27,
 11-14
 Relay node parameter, 13-6
 Remote FTP server, 3-9, 5-6, 5-8,
 B-9
 program messages, B-9
 RENAME command (PRIMOS
 TCP/IP), 3-28, 4-16
 RESUME command (PRIMOS), 11-3
 RETRIEVE command (PRIMOS
 TCP/IP), 3-28, 4-16
 Retrieving a file, 3-18
 RMD command (PRIMOS TCP/IP),
 5-6
 RMDIR command (PRIMOS
 TCP/IP), 3-28, 4-16
 RNFR command (PRIMOS TCP/IP),
 5-7, B-10 to B-11
 see also: RNTO command
 RNTO command (PRIMOS TCP/IP,
 5-7, B-10 to B-11
 see also: RNFR command
 ROUTE.CACHE file, 13-3, 13-18
 Runfiles
 see: PRIMOS TCP/IP runfiles

S

SAD directory, 5-3
 Sample socket programs
 client, 8-9, 10-1
 datagram, 8-19
 server, 8-10, 10-4
 stream, 8-8
 Search rules, 11-17
 include, 8-28
 SEMAPHORE_ALLOCATION_
 FILE file, 11-9
 SEND command (PRIMOS TCP/IP),
 3-14, 3-28, 4-17
 SERVER_FTP_LOG file, 14-10,
 16-3
 SET_SEARCH_RULES command
 (PRIMOS), 11-17
 SHOW command (PRIMOS TCP/IP),
 3-23, 4-5
 SIGCONTROLLERDOWN signal,
 8-22 to 8-23, 9-10
 Signals, 8-22
 SIGPIPE signal, 8-14, 8-22 to 8-23,
 9-10

- SIGSOCKETACCEPTED signal, 8-23, 8-26, 9-7
- SIGSOCKETEXCEPTION signal, 8-23, 8-25, 9-24
- SIGURG signal, 8-18, 8-23
- Simple Mail Transfer Protocol (SMTP), 1-2, 1-5
- SMTP_LOG file, 14-11, 16-3
- SMTP.RUN file, 11-10
- SMTP_SENDER0, 6-4, 12-18, 14-2, 14-13
- SMTP_SERVERnn, 6-4, 14-2, 14-13
- SO_BUFSIZE socket option, 9-32, 9-48
- Sockaddr structure, 8-3
- Sockaddr_in structure, 8-4
- SOCK_DGRAM sockets, 8-3
- Socket layers, 8-2
- Socket names, 8-3
- Socket options, 9-31, 9-47
- Socket programming, 1-6
 - blocking sockets, 9-16
 - closing sockets, 8-19, 9-22
 - compiling an application, 8-29
 - creating sockets, 8-14, 9-51
 - displaying error messages, 8-28
 - establishing connections, 8-15, 9-16, 9-23
 - examples, 8-8, 8-19, 10-1
 - getting options, 9-31
 - linking an application, 8-29
 - naming sockets, 8-15, 9-19
 - privileged connections, 8-13, 9-10, 9-13
 - running an application, 8-30
 - setting options, 8-15, 9-46 to 9-47
 - setting up the environment, 8-12, 9-13
 - sharing sockets, 8-26, 9-7*see also*: PRIMOS socket library
- Socket subroutines
 - see*: PRIMOS socket routines
- Socket types
 - SOCK_DGRAM, 8-3
 - SOCK_STREAM, 8-3
- SOCKET.BIN file, 11-12
- SOCKET.H.INS.CC file, 8-27, 11-14
- SOCKET_IX.MAP file, 11-12
- SOCKET_IX.RUN file, 11-12, 11-17
- SOCKET.MAP file, 11-12
- SOCKET.RUN file, 11-12, 11-17
- SOCK_STREAM sockets, 8-3
- SO_KEEPALIVE socket option, 9-31, 9-47
- SO_LINGER socket option, 9-32, 9-48
- SOL_SOCKET socket level, 8-30, 9-30, 9-46
- SO_REUSEADDR socket option, 9-31, 9-47
- START_DSM command (DSM), 14-16 to 14-17
- START_FTP_SERVER command (PRIMOS TCP/IP), 16-3
 - format of, 14-10
 - in the PRIMOS.COM1 file, 14-17
- START_FTP_SERVER.HELP file, 11-11
- START_FTP_SERVER.RUN file, 11-8
- START_MAILER.HELP file, 11-11
- START_MAILER.RUN file, 11-8, 11-13
- START_NTS command (PRIMOS), 14-3, 14-9, 14-16 to 14-17
- START_SMTP command (PRIMOS TCP/IP), 14-11, 14-16
- START_SMTP.HELP file, 11-11
- START_SMTP.RUN file, 11-8
- START_TCP/IP command (PRIMOS TCP/IP), 14-3, 14-9, 14-12, 14-16 to 14-17
- START_TCP/IP.HELP file, 11-11
- START_TCP/IP_LOG file, 14-9, 16-3
- START_TCP/IP.RUN file, 11-8
- Stations, defined, 1-3
- STATUS COMM command (PRIMOS), 12-11
- STATUS command (PRIMOS TCP/IP), 2-7, 3-13, 3-23, 4-5
- STATUS NTS command (PRIMOS), 14-13
- STDIO.H.INS.CC file, 8-27
- STOP_MAILER command (PRIMOS TCP/IP), 14-16
- STOP_MAILER.HELP file, 11-11
- STOP_MAILER.RUN file, 11-8, 11-13
- STOP_TCP.HELP file, 11-11
- STOP_TCP/IP command (PRIMOS TCP/IP), 12-15, 14-15
- STOP_TCP/IP_LOG file, 14-15
- STOP_TCP/IP.RUN file, 11-8
- STORE command (PRIMOS TCP/IP), 3-28, 4-17
- STREAM mode, 4-13, 5-6, B-12
- STRU command (PRIMOS TCP/IP), 5-7
- Structure
 - see*: File structure
- STRUCTURE command (PRIMOS TCP/IP), 3-14, 3-28, 4-5, 4-17
- Subnet bits
 - see*: Subnet mask
- Subnet mask, D-2, 12-13, 15-9, 15-10
 - examples of, 12-14
 - subnet bits, 12-14
- Subnet number, 15-9
- Subnets
 - see*: Subnetworks
- Subnetting
 - see*: Subnetworks
- Subnetworks, 12-12, 15-1, 15-12
 - definition of, 12-12
 - examples of, 12-13, 15-7
 - host addresses, 12-13
 - implementation, 12-13
 - subnet mask, 12-13, 15-9
 - subnet number, 15-9
- Sub-site names, 13-6
- Synchronizers, 8-25 to 8-26, 9-11, 9-14
- SYS.COM directory, 8-27, 11-3, 11-7, 11-13
 - INET.H.INS.CC file, 11-13
 - IN.H.INS.CC file, 11-13
 - IOCTL.H.INS.CC file, 11-14
 - NETDB.H.INS.CC file, 11-14
 - REDEFINE.H.INS.CC file, 11-14
 - SOCKET.H.INS.CC file, 11-14
 - TCP/IP_ERROR.H.INS.CC file, 11-14
 - TCP/IP_GLOBAL.H.INS.CC file, 11-14
 - TCP/IP_TIME.H.INS.CC file, 11-14

SYSNAM directive, 12-23, 15-2, 15-13
 SYSOVL directory, 11-3, 11-7, 11-14
 TUP\$ _ERROR_TABLE file, 11-14
 System configuration file, modifying, 12-17

T

TCP

see: Transmission Control Protocol
 TCP protocols, 14-7

.TCPFTP\$ ACL group, 1-5, 3-3, 5-1

TCPFTP_SERVER nn process, 14-2

TCPFTP_SERVER_PHANTOM Mnn
 phantom, 14-2, 14-11 to 14-12

TCPFTPSPRUN file, 11-10

TCPFTPSS.RUN file, 11-10

TCP/IP directory, 11-3

 TCP/IP.INSTALL.CPL file, 11-3, 11-16

TCP/IP* directory, 11-3, 11-7, 11-9, 11-16, 15-1, 15-5, 15-13

 HOSTNAME_CACHE file, 12-16

 HOSTS.TEMPLATE file, 11-9

 SEMAPHORE_ALLOCATION_FILE file, 11-9

 TCP/IP.INSTALL_ACL.CPL file, 11-9

 TCP/IPHTU.RUN file, 11-9

 THISHOST.TEMPLATE file, 11-9

TCP/IP*>COMI directory, 11-9

TCP/IP_DUMP.RUN file, 11-10

TCP/IP_ERROR.H.INS.CC file, 8-27, 11-14

TCP/IP_GLOBAL.H.INS.CC file, 11-14

TCP/IP_INIT_HOSTS command
 (PRIMOS TCP/IP), 12-15, 14-14

TCP/IP.INSTALL_ACL.CPL file, 11-9, 11-15

TCP/IP.INSTALL.CPL file, 11-3, 11-16

 -PRIMENET option, 11-3

TCP/IP*>ISC directory, 11-9

TCP/IP*>LOG_FILES directory, 11-7, 11-9, 14-9, 16-2

TCP/IP*>LOG_FILES>ARCHIVE
 directory, 11-9

TCP/IP_MANAGER process, setting
 NPUSR directive for, 12-19

TCP/IP_MANAGER.RUN file, 11-10

TCP/IP_MES.OUT file, 11-10

TCP/IP*>MESSAGE_TEXT
 directory, 11-7, 11-10

 TCP/IP_MES.OUT file, 11-10

 TCP/IPHTU_MES.OUT file, 11-10

 TEL_MES.OUT file, 11-10

TCP/IP*>RUN directory, 11-7, 11-10

 SMTP.RUN file, 11-10

 TCPFTPSPRUN file, 11-10

 TCPFTPSS.RUN file, 11-10

 TCP/IP_DUMP.RUN file, 11-10

 TCP/IP_MANAGER.RUN file, 11-10

TCP/IP.RUNO file, 11-12

TCP/IP

see: PRIMOS TCP/IP

TCP/IP*>SMTP directory, 11-7, 11-10

TCP/IP*>SMTP>INCOMING
 directory, 11-10

TCP/IP*>SMTP>OUTGOING
 directory, 11-11

TCP/IP_TIME.H.INS.CC file, 8-27, 11-14

TCP.INSTALL_ACL.CPL file, 11-15

TCP/IPHTU

see: Host Table Utility

TCP/IPHTU.HELP file, 11-11

TCP/IPHTU_MES.OUT file, 11-10

TCP/IPHTU.RUN file, 11-9, 12-16

TCPIP_MANAGER process, 14-3, 14-12

 description of, 14-3

 logging out, 14-16

TEL_MES.OUT file, 11-10

TELNET, 1-2, 1-8, 11-3, 12-19

 Command mode, 2-1, 2-3

 commands, 2-4

 default escape character, 2-4

 enabling EDIT_CMD_LINE, 2-4

 escape character, 2-3

 Input mode, 2-2

 invoking, 2-1

 Server TELNET, 1-4

 User TELNET, 1-4

TELNET command (PRIMOS
 TCP/IP), 2-1

TELNET commands, 2-4

 CLOSE, 2-6

 DEBUG, 2-7

 ESCAPE, 2-5

 HELP, 2-7

 OPEN, 2-6

 OPTIONS, 2-7

 QUIT, 2-7

 STATUS, 2-7

TELNET.HELP file, 11-11

TELNET.RUN file, 11-8

THISHOST file, 12-1, 12-11

 adding subnet mask to, 12-14

 creating, 12-11

 examples of, 15-2, 15-6, 15-9, 15-13

THISHOST.TEMPLATE file, 11-9

Ticks, F-1

Transmission Control Protocol (TCP),
 1-1 to 1-2, 5-6, 14-7

 and socket types, 8-3

Transport-scan-interval parameter,
 13-8

TUP\$accept(), 8-17, 9-16

TUP\$bind(), 8-15, 9-19

TUP\$close_socket(), 8-19, 9-22

TUP\$close_socket_env(), 8-19, 9-5

TUP\$connect(), 8-15, 9-23

TUP\$ _ERROR_TABLE file, 11-14

TUP\$getalladdrs(), 9-54

TUP\$getbroadcastaddr(), 9-55

TUP\$getdevaddrs(), 8-12, 9-57

TUP\$gethostbyaddr(), 9-60

TUP\$gethostbyname(), 9-63

TUP\$gethostname(), 9-65

TUP\$getpeername(), 9-26

TUP\$getsockname(), 9-28

TUP\$getsockopt(), 9-30

TUP\$inet_addr(), 9-66

TUP\$inet_lnaof(), 9-67

TUP\$inet_makeaddr(), 9-68

TUP\$inet_netof(), 9-69

TUP\$inet_network(), 9-70

TUP\$inet_ntoa(), 9-71

TUP\$ioctl(), 9-33

TUP\$listen(), 8-16, 9-35
 TUP\$pass_socket(), 8-26, 9-7
 TUP\$perror(), 9-37
 TUP\$receive_passed_socket(), 8-26, 9-9
 TUP\$recv(), 8-17, 9-38
 TUP\$recvfrom(), 8-17, 8-20, 9-38
 TUP\$reset_socket_env(), 8-22, 9-10
 TUP\$select(), 9-41
 TUP\$send(), 8-17, 9-43
 TUP\$sendto(), 8-17, 8-20, 9-43
 TUP\$setsockopt(), 9-46
 TUP\$setup_socket_env(), 8-13, 9-13
 TUP\$shutdown(), 9-49
 TUP\$socket(), 8-14, 9-51
 TYPE command (PRIMOS TCP/IP), 3-13, 3-28, 4-5, 4-17, 5-8
 description, 12-9
 example of in HOSTS.TXT, 12-9
 Wildcard names, 3-21, 4-11
 used with LS command, 4-11
 used with MGET command, 3-21, 4-11
 used with MPUT command, 3-17, 4-13
 used with NLST command, 4-14
 Wildcarding, 4-5, 5-5 to 5-6
 Workstation/System Interconnect 300, 1-1, 5-1
 WSI300, 3-2
 .WSIFTP\$ ACL group, 3-2, 5-1

U

.UBI\$ ACL group, 8-13, 9-10, 9-13, B-2, B-5
 UBI_SERVER process, 14-12
 UDP
 see: User Datagram Protocol
 ULD.LOG file, 16-3
 UNASSIGN command (PRIMOS), 11-4
 Unconfigured nodes, 12-21, 15-6
 UNIX operating system, 1-5
 Upline dump default file, 16-1
 Upline dumping, 12-22
 UP_LINE_DUMP*>LAN300
 directory, 16-1
 USER command (PRIMOS TCP/IP), 3-9, 3-12, 3-28, 4-18, 5-6, B-12 to B-13
 User Datagram Protocol (UDP), 1-2, 8-1
 and socket types, 8-3
 User IDs, 1-5, 3-4, 5-8
 User Profile Database, 11-15
 User TELNET, 1-4, 2-1, A-1

W

Wide-area-interval parameter, 13-9
 Wide-area-retry parameter, 13-9
 Wide-area-search parameter, 13-10
 Wildcard gateways, 12-9